لغة PHP لمبرمجي الويب

لغة

PHP

لمبرمجي الويب

تأليف

المهندس فهمي الصيرفي

لغة PHP لمبرمجي الويب

تأليف: المهندس فهمي الصيرفي

الطبعة الأولى: ٢٠٠٩.

عدد النسخ: ١٠٠٠ نسخة.

جميع العمليات الفنية والطباعية تمت في:

دار ومؤسسة رسلان للطباعة والنشر والتوزيع

يطلب الكتاب على العنوان التالي

دار رسلان

للطباعة والنشر والتوزيع

سوريا ـ دمشق ـ جرمانا

هاتف:٥٦٢٧٠٦٠ ١١ ٠٠٩٦٣

فاكس: ٥٦٣٢٨٦٠ ١١ ٠٠٩٦٣

ص. ب: ٢٥٩ جرمانا

الفصل الأول
مقدمة حول لغة PHP

تتميز لغة PHP بالكثير من الخصائص التي جعلتها الخيار الأمثل لمبرمجي الويب في العالم:

١- السهولة

تعتبر لغة PHP من أسهل لغات البرمجة تعلماً، فهي تريحك من جميع تعقيدات إدارة الذاكرة وتعقيدات معالجة النصوص الموجودة في C من جهة، والكثير من الضعف الموجود في بينية وتصميم لغة البرمجة Perl من جهة أخرى.
تمتلك لغة PHP بنية وقواعد ثابتةٍ وواضحةٍ جداً، معظم قواعد اللغة مأخوذة من كل من C وJava و Perl لصنع لغة برمجة عالية السهولة والسلاسة دون فقدان أي من القوة في اللغة، يفيدك ذلك إذا كنت تعلم أي شيء عن لغات البرمجة الأخرى مثل Visual Basic أو C أو Java حيث ستجد دائماً بأنك تفهم مواد الكتاب بسرعة، وستكتشف كيف تقوم PHP بتسهيل أصعب الأمور وإذلال العقبات التي تواجه المبرمج حتى يتفرغ تماماً للإبداع فقط، كل ما تفكر به تستطيع تنفيذه بلغة PHP.

٢- السرعة

لغة PHP من اللغات المعروفة بسرعتها العالية في تنفيذ البرامج، وخاصة في الإصدار الرابع من المترجم، حيث تمت كتابة مترجم PHP من الصفر ليعطي أداء في منتهى الروعة، كما أن لغة PHP مصممة أصلاً كنواة لمترجم، بحيث يمكن أن تضع هذه النواة في عدة قوالب أو أغلفة لتعمل مع التقنيات المختلفة، فيمكنك تشغيل مترجم PHP كبرنامج CGI مثلاً، ولكن الأفضل هو إمكانية تركيب مترجم PHP على مزود IIS في صورة وحدة إضافية تضاف إلى المزود عن طريق دوال ISAPI، وتوجد نسخة أخرى منه تركب على مزود Apache أيضاً في صورة وحدة خارجية، وتوجد أيضاً نسخة مخصصة للدمج مع شفرة مزود Apache بحيث تصبح جزءاً من برنامج Apache نفسه، وهي الطريقة الأكثر استخداماً

الآن في مزودات الويب التي تعمل على أنظمة UNIX وهي الطريقة التي تعطي أفضل أداء لمترجم PHP، حيث يصبح المترجم جزءاً من المزود، وبالتالي فإنه سيكون محملاً في الذاكرة بانتظار صفحات PHP ليقوم بترجمتها وعرضها للزوار مباشرة دون التأخير الإضافي الذي تتطلبه برامج Perl/CGI مثلاً حيث يجب أن يتم تشغيل مترجم Perl مع كل زيارة للصفحة لترجمة الصفحة، ثم يتم إغلاق المترجم، ثم استدعاءه مجدداً عند الزيارة الثانية وهكذا، وهذا يشكل فارقاً كبيراً في المواقع ذات الضغط العالي بالذات، ويكون استخدام PHP حلاً أفضل بكثير.

٣- المزايا

يأتي مترجم PHP لوحده محملاً بعدد هائل من الدوال الجاهزة للاستخدام في جميع المجالات، من دوال المعالجة الرياضية والحسابية إلى دوال الوصول إلى قواعد البيانات ومزودات FTP، توفر لك دوال PHP مثلاً وصولاً إلى مزودات البيانات MySQL و PostgreSQL و MS SQL و Oracle وغيرها من مزودات قواعد البيانات، وهنالك أيضاً مجموعة من الدوال لمعالجة ملفات XML، ودوال أخرى لإرسال واستقبال الملفات عن بعد باستخدام بروتوكول FTP، وهنالك مجموعة من الدوال لمعالجة وإنتاج الصور ديناميكياً وملفات Flash ديناميكياً، ناهيك عن جميع الدوال الخاصة بمعالجة النصوص والمصفوفات.

٤- التوافقية

كما قلنا سابقاً، فعلى الرغم من أن هنالك الكثير من نسخ PHP التي يعمل كل منها في بيئة مختلفة، إلا أنها جميعاً تشترك في النواة الأصلية التي تقوم بالمعالجة الحقيقة لملفات PHP لذا فإن جميع مترجمات PHP تتصرف بنفس الطريقة فيما يتعلق بتنفيذ السكريبتات، فإذا كان السكريبت الذي عملته يعمل على نظام Windows مع مزود IIS فيجب أن يعمل دون الحاجة لأية تغييرات عند نقله إلى مزود Apache، بالطبع تظل بعض الأمور البسيطة جداً التي توفرها بعض المزودات دون غيرها، ولكن جميع البرامج التي كتبتها منذ أن بدأت تعلمي للغة إلى الآن تعمل على جميع المزودات دون الحاجة لأي تغييرات، إضافة إلى ذلك فإن التغييرات

التي حدثت باللغة الأساسية من الإصدار الثالث إلى الرابع قليلة جداً، وأغلب التغييرات كانت في البنية التحتية للمترجم.

٥- الحماية

يوفر PHP الكثير من المزايا المتقدمة، ولكنه يوفر لك الطرق المناسبة لوضع الحدود على هذه المزايا، فيمكنك التحكم بعدد الاتصالات المسموحة بقاعدة البيانات مثلاً، أو الحجم الأقصى للملفات التي يمكن إرسالها عبر المتصفح، أو السماح باستخدام بعض الميزات أو إلغاء استخدامها، كل هذا يتم عن طريق ملف إعدادات PHP والذي يتحكم به مدير الموقع.

٦- قابلية التوسع

يمكنك توسعة مترجم PHP بسهولة وإضافة الميزات التي تريدها إليه بلغة C، وحيث إن الشيفرة البرمجية للمترجم مفتوحة فإنك تستطيع تغيير ما تريده مباشرة لتحصل على النسخة التي تناسبك من المترجم، ويمكنك أيضاً عمل الوحدات الإضافية التي تركب على المترجم لزيادة ميزاته والوظائف المبيتة فيه، وقد قام فريق تطوير مترجم PHP مسبقاً بعمل هذه المهمة وتحويل كمية ضخمة من المكتبات المكتوبة بلغة C إلى مكتبات مخصصة لتضاف إلى المترجم، ومنها حصلنا على جميع الميزات التي تحدثنا عنها مثل الوصول إلى قواعد البيانات ومعالجة ملفات XML.

تاريخ PHP

بدأت PHP كمكتبة من الدوال تضاف على لغة Perl لتسهل عمل برامج CGI بلغة Perl، وبعد أن تلقى Rasmus Lerdof بعض الاقتراحات بتحويلها إلى مترجم بسيط، قام بعمل ذلك المترجم وطرحه على الإنترنت وسماه PHP أو Personal Home Pages أي الصفحات الشخصية، فقد كان عبارة عن نسخة مصغرة من Perl مع بعض الميزات الإضافية للويب، ثم أضاف إليه دعماً لنماذج HTML وسماه PHP2/FI، فقام مجموعة من المبرمجين بالعمل على مترجم

PHP وأضـافوا إليـه واجهـة تطبيقـات برمجيـة API لتسهيل عمليـة توسـعته فأصبـح لدينا PHP 3 ، بعـد فـترة مـن الـزمن قامـت شـركـة Zend للتقنيات بعمـل مترجمهـا الخـاص للغـة والـذي سمـي zend أيضاً ، وقـد اتـصف هـذا المترجـم بالـسرعة العاليـة وقدراتـه المحسنة ، وجمـع مـع مكتبـات PHP الأخـرى لتكـوين نـواة المترجـم PHP ، مترجم PHP الآن مقسم إلى قسمين: المترجم zend ويتم تطويره على مزودات CVS الموجـودة في موقـع zend والقـسم الثـاني يـسمى PHP وهـو عبـارة عـن المكتبـات والـدوال الأساسـية الـتي تـأتي مـع البرنـامج، يقـوم مترجـم zend بقـراءة الملفـات ومعالجتهـا والتعامـل مـع المـتغيرات وتنفيـذ البرنـامج وتـوفير واجهـة تطـوير للتطبيقـات API لتوسعة اللغة ، أما PHP فتحتوي الآن على مكتبات مكتوبـة بلغـة C ومتوافقـة مـع واجهـة التطبيقـات الـتي يوفرهـا مترجـم zend ، وبالتـالي يعمـل القـسمان معـا لتكوين مترجم PHP ، وعندما تزور موقـع PHP الرسمـي الآن وتحـصل علـى مترجـم PHP جاهز أو تحصل على الشيفرة البرمجية الخاصـة بـك ، فإنـك تحـصل علـى كـل من مترجم zend ومكتبات PHP معاً.

تطور PHP تطوراً مفاجئـاً في الفـترة الأخيرة ، وتـشير إحـصائياً Net Craft إلى أن مترجـم PHP هـو أكثـر وحـدات مـزود Apache انتـشاراً علـى الإنترنـت، كمـا أن مترجم PHP مركب على حوالي مليوني مزود ويب على الإنترنت.

بنية ملفات PHP

ملفـات PHP هـي ملفـات نـصية بـسيطة ، تـشبه في تركيبهـا ملفـات ASP وملفـات HTML بشكل عام، يتكون ملف PHP من قسمين، قسم HTML وقسم PHP ، الملـف بالـصورة الطبيعيـة عبـارة عـن ملـف HTML عـادي، ولكنـك تـستطيع تحديـد أجـزاء معينـة مـن الملـف ليخـرج فيهـا الملـف مـن وضـعية HTML إلى وضـعية PHP ، لإخراج الملف إلى وضعية PHP توجد عدة طرق:

١ - استخدام زوج الوسوم <?php و ?> كالتالي:

```
<?php
echo 'This is PHP output!';
?>
```

٢ - استخدام زوج المختصر <? و ?> وهو يستخدم بنفس الطريقة السابقة ولكنه يكون بدون الكلمة php في وسم البداية، هذا النوع من الوسوم يحتاج إلى كمية أقل من الكتابة بالطبع، ولكنه يتعارض مع وسوم xml، لذا يقوم البعض بإغلاق ميزة الوسوم القصيرة حتى لا يحصل هذا التعارض (يمكنك إغلاق هذه الميزة بسهولة عن طريق ملف إعدادات PHP).

٣ - استخدام زوج الوسوم ASP، وهو من اسمه زوج الوسوم المستخدم في ملفات ASP وهما <% و %>، ميزة وسوم ASP لا تكون فعالة بشكل قياسي ولكنك تستطيع تفعيلها عن طريق ملف إعدادات مترجم PHP.

٤ - الطريقة الأخيرة هي استخدام زوج الوسوم التالي:

```
<script language="php" >
echo 'This is PHP output!';
</script>
```

ولكن هذه الطريقة غير مستخدمة الآن، حيث إنها تصعب عملية التمييز بين شيفرات PHP وباقي ملف HTML، وكذلك بالنسبة لبرامج كتابة ملفات HTML التي تعطي تلوينا للشيفرة فأغلبها لا يتعرف على هذا النوع من الشيفرة ويعتبره جزءاً من ملف HTML الاعتيادي.

أفضل الطرق السابقة للتحويل إلى وضعية PHP هو استخدام زوج الوسوم الأول بالطبع، حيث إنه الأكثر استخدامها، ولا يحتوي على أية تعارضات كما إنه يعمل على جميع مترجمات PHP مهما كانت إعداداتها، ولهذا السبب سنستخدمها في جميع الأمثلة التي ستجدها في هذا الكتاب.

كتابة ملفات PHP

ملفات PHP هي ملفات نصية بسيطة تماماً كما هي ملفات HTML، يمكنك كتابة سكريبت PHP بأي برنامج كتابة نصوص يتيح لك كتابة الملفات النصية البسيطة Plain Text مثل Notepad على النظام ويندوز، ولكن أغلبية مبرمجي PHP يستخدمون أدوات أخرى تسهل عليهم عملية البرمجة عن طريق تلوين الشيفرات البرمجية، وتسهل عملية البحث عن الملفات واستبدال المقاطع من عدة ملفات في نفس الوقت، مثل HomeSite، على الرغم من أنك لن تحتاج إلى الكثير من هذه الميزات إلا أن استخدام Notepad في عمل ملفات PHP يعتبر أمراً صعباً جداً وخاصة في الملفات الضخمة حيث أن Notepad لا تتيح فتح الملفات الكبيرة، والمشكلة الأكبر هي أنها لا توفر ترقيماً للأسطر، فإذا ظهرت لك رسالة الخطأ تشير إلى وجود خطأ في السطر ٥٣ فلن تستطيع معرفة السطر المطلوب في Notepad إلا إذا قمت بالعد يدوياً من السطر الأول وحتى ٥٣.. حسناً ماذا لو كان الخطأ في السطر ٦٥٢، يمكنك البدأ بكتابة سكريبتاتك بالبرنامج المتوفر الآن إلى أن تحصل على برنامج آخر، يمكنك بالطبع فتح ملفاتك بأي محرر نصوص، فإذا كتبتها باستخدام Notepad فهذا لا يعني بأنك ملزم باستخدام Notepad في جميع ملفاتك أو حتى في هذا الملف.

لعمل ملف PHP الآن قم بفتح محرر النصوص الذي اخترته وابدأ بكتابة الصفحة التي تريدها، ولا تنس إحاطة شيفرات PHP بالوسوم الخاصة بها، ثم احفظ الملف في أي مكان في دليل مزود الويب الخاص بك وأعطه الامتداد المناسب.php أو.php3 حسب إعدادات مزودك، ثم قم بزيارة الصفحة باستخدام المتصفح وستجد الصفحة وقد تمت ترجمتها وعرضها عليك.

تذكر بأنك يجب أن تزور الصفحة مروراً بمزود الويب، ولا يمكنك عرض الصفحة عن طريق فتحها كملف خارجي، على سبيل المثال، إذا كان الدليل الجذري لصفحات مزودك هو: C:\httpd\

وقمت بعمل صفحة أسميتها test.php في ذلك الدليل، يجب أن تقوم الآن بتشغل مزود الويب وزيارة الصفحة على العنوان http://localhost/test.php، إذا قمت باستخدام الأمر Open من القائمة File في المتصفح لفتح الملف C:\httpd\test.php فلن ترى صفحة PHP مترجمة، وسترى شيفرة PHP فقط.

تدريب

قم بتنفيذ ملف PHP التالي:

```
This is the normal html page.<br>
<?php
echo "This is inside PHP<br>";
echo "Hello World!<br>";
?>
```

ما الذي تشاهده عند تنفيذ البرنامج السابق؟ من المفترض أن تشاهد الخرج التالي:

```
This is the normal html page.
This is inside PHP
Hello World!
```

ها قد انتهيت من كتابة برنامجك الأول بلغة PHP، لا تقلق إذا لم تفهم أي شيء فيه، سنتعلم الآن كيفية استخدام المتغيرات والعبارات بلغة PHP.

لنكتب سكريبتاً بسيطاً (فاتح شهية):

```
<”html dir = “rtl>
الوطن العربي
؟>
Echo (”من المحيط إلى الخليج“)
<؟
<html/>
```

قم بحفظ الملف باسم echo.php

ستعرض علينا عبارة مكتوب فيها

الوطن العربي من المحيط إلى الخليج

شي بسيط أليس كذلك ؟

يتكون كود الـphp من نصوص و كود و علامات ولغة html وقد لاتحتوي على نصوص html.

لكي يعمل الكود يجب أن يكون امتداد الملف php أو بأي امتداد من امتدادات الـphp

مثلاً php3 و phtml

عندما تطلب صفحة في الإنترنت فإنك تجري اتصالاً مباشراً مع السيرفر هذه العملية تدعي request للسيرفر (يعني طلبية للسيرفر) يقوم السيرفر بتفسير طلبك والبحث عن الصفحة المطلوبة ويرسل إليك الصفحة المطلوبة كجزء مما يسمي

response (استجابة) لمستعرض الإنترنت لديك يقوم بعدها المتصفح لديك بأخذ الكود الذى أرجع إليه ويقوم بتجميعه (compile) لكي يصبح صفحة صالحة للعرض هذه العملية التي حصلت تشبه نظرية العميل للخادم(client to server) بحيث أن المتصفح هو العميل والخادم هو السيرفر.

الخادم يقوم بعملية تخزين وترجمة وتوزيع البيانات بينما يقوم العميل (مستعرض الإنترنت لديك) بالعبور إلى السيرفر وإحضار البيانات

الفصل الثاني
بروتوكولات الإنترنت

لانريد هنا أن نذهب إلى التكلم عن تاريخ انترنت العتيق، النقطة المهمة هي الشبكة المربوطة بنقاط nodes الانترنت صممت لكي تقوم بالحفاظ على المعلومات لكي يتم نقلها من مكان إلى آخر وهي تستخدم مجموعة من البروتوكولات مثل Tcp/Ip لكي يتم نقل البيانات عبر الشبكة.

بروتوكول Tcp / Ip

من مميزات هذا البروتوكول أنه باستطاعته إعادة تمهيد طريقه للبيانات إذا حدث خلل في نقطة أو مكان أثناء نقلها ويتم ذلك بسرعة شديدة. عندما يطلب المستخدم من المستعرض أن يجلب له صفحة من الإنترنت فإن المستعرض يجلب هذه الأوامر باستخدام بروتوكول يدعي بروتوكول التحكم في نقل البيانات TCP هذا البروتوكول هو بروتوكول نقل للبيانات وهو يضمن أن البيانات قد تم إرسالها ووصولها بشكل صحيح.

قبل أن يتم إرسال البيانات عبر الشبكة يجب عنونتها والبروتوكول الذي يقوم بعنونة البيانات يدعي HTTP يقوم هذا البروتوكول بوضع عنونة للبيانات لكي يعرف البروتوكول TCP أين سينقل البيانات (فهو لايستطيع نقل البيانات إذا لم يكن لها هدف أو مكان) يستخدم البروتوكول HTTP عن طريق الويب في عملية نقل البيانات من كمبيوتر إلى آخر عندما ترى الصفحة متبوعة بـhttp:// فإنك تعلم مباشرة أن الإنترنت تستخدم البرتوكول HTTP لإحضار هذه الصفحة يمكنك أن تاخذ صورة بأن الـTCP عبارة عن ساعي البريد الذي يقوم بإيصال رسالة، هذه الرسالة فيها طابع بريد وعنوان وهو مانسميه بالـHTTP.

يتم تمرير الطلب من المستعرض إلى ملقم أو سيرفر الويب وهو مايعرف بـ HTTP request ويقوم السيرفر برؤية مستودع البيانات لديه لكي يحصل على البيانات المطلوبة فإذا وجد الصفحة في المستودع قام بإرسالها على شكل حزم إلى الجهة التي قامت بالطلب باستخدام بروتوكول TCP ويعنون هذه الحزم لمستعرض الإنترنت لديك باستخدام بروتوكول http (ننبه دائماً إلى أنه يرسلها على شكل حزم لكي تعرف السبب عند عدم ظهور صفحة ويب كاملة أن هناك حزمة لم ترسل بشكل جيد) ولكن إذا لم يجد السيرفر الصفحة المطلوبة فإنه يقوم بإرسال صفحة تحتوي على رسالة خطأ ٤٠٤ وهذه الصفحة التي أرسلت من ملقم الويب إلى المستعرض لديك تسمي HTTP response.

بروتوكول الـ HTTP

رغم ما أخذناه من معلومات كثيرة وقصص كثيرة تشبه قصص ألف ليلة وليلة أو حكايات الأطفال إلا أنه رغم ذلك يفوتنا الكثير من التفاصيل في هذا الموضوع لذلك دعونا نغوص قليلاً في تفاصيل بروتوكول HTTP بشكل خاص.

عندما تقوم بعملية طلب لصفحة من السيرفر هناك أمور إضافية ترسل مع عملية الطلب http request غير ال URL وهي ترسل كجزء من http request.
نفس الموضوع مع الhttp response هناك أمور أخرى تصل معه كجزء منه.

الكثير من هذه المعلومات تولد تلقائياً في رسالة الـHTTP ولايقوم المستخدم بالتعامل معها مباشرة، إذن لايحتاج أن تقلق نفسك بشأن هذه المعلومات إذا أنت لم تنشأها في الأصل ويجب أن تأخذ أيضاً في معلوماتك أن هذه المعلومات ترسل كجزء من الـHTTP request والـHTTP response لأن سكريبت الـPHP الذي نصنعه يمنحنا تحكماً إضافياً بهذه المعلومات.

كـــل رســـائل الـــHTTP تأخــذ تنــسيقاً معينـــاً ســـواء كانـــت Request أو Response. نستطيع أن نقوم بتقسيم هذا التنسيق إلى ثلاثة أقسام:

١ - Request/response line.

٢ - Http header.

٣ - Http body.

المحتـــوي مـــن هـــذه الأشـــياء الثلاثـــة يعتمـــد علـــى نـــوع الرســـالة إذا كانـــت HTTp Request أو HTTP response لذلك سنتكلم عنهم بتعمق أكثر.

Http Request

يجـب أن يحتـوي الـrequest علـى الأقـل الـline request (سـطر الطلـب) والـHOST.

يرسـل مـستعرض الإنترنـت طلبيـة (HTTP request) إلى ملقـم الويـب تحتـوي علـى التالي:

١- The Request Line

الـسطر الأول مـن كـل طلبيـة (http request) هـي Request Line الـذي يحتوي على ثلاثة أنواع من المعلومات:

أ - أمر HTTP وهو مايعني بـ method.

ب - المـسار مـن الـسيرفر إلى المـصادر المطلوبـة (صـفحات الإنترنـت) المطلوبـة مـن قبـل العميل (المستعرض)

جـ - إصدار الـHTTP.

إذن كمثال على الـ Request Line أنظر إلى السطر التالي:

GET /testpage.htm HTTP/1.1

الـmethod يخبر السيرفر كيف يتعامل مع الطلب، هناك ثلاثة أنواع شائعة من الـ method.

٢- HTTP Header

البت الثاني من المعلومات هو الهيدر HTTP Header.الذي يحتوي على تفاصيل أو وثائق عن العميل مثل نوع المتصفح (نتسكيب أو إكسبلور) الذي قام بطلب الصفحة والوقت والتاريخ والإعدادات العامة.

الـHTTP Header يحتوي على معلومات نستطيع تقسيمها إلى ثلاث فئات وهي:

أ - عامة GENERAL: تحتوي معلومات إما عن العميل أو السيرفر ولاتخصص إلى فرد أو مجموعة.

ب - شخصية Entity: تحتوي على معلومات عن البيانات التي أرسلت بين المتصفح والسيرفر.

جـ - مطلوبة Request: تحتوي على بيانات عن إعدادات العميل والأنواع المختلفة المقبولة من البيانات.

وهذا مثال:

Accept: ❖ / ❖

Accept language: Arabic-KSA.
Connection: Keep –Alive.
Host: http://www.arabbuielder.com
Referer: http://www.arabbuielder.com/index.php?something=132
User –Agent:Iexploer (win98؛.........)

مثلما ترى الـ HTTP Header عبارة عن إعداد يتكون من عدة سطور كل سطر يحتوي على قيم معينة.

هناك عدة سطور تشكل الـ HTTP header وأكثرها اختياري، يقوم الـHTTP بالإخبار عن انتهاء معلومات الـ header بترك سطر فارغ (وهذا يكون في الـHTTP1.1).

٣- The HTTP Body:

إذا تم استخدام الأمر POST في الـHTTP Request Line عندها يقوم الـ HTTP بطلب المعلومات التي أرسلت في الـbody إلى السيرفر.

Http Response

يرسل من السيرفر إلى المستعرض ويحتوي على ثلاثة أشياء:

١- the Response Line.

٢ - http header .

٣ - Http Body .

١ - The Response Line

الـ response line يحتوي فقط على نوعين من المعلومات:

١ - رقم إصدار الـHTTP.

٢ - شيفره أو كود الـhttp request التي تقوم بتحديد إذا كان الـrequest ناجحاً أم فاشلاً.

مثال:

HTTP/1.1 200 OK

في هذا المثال يقوم الـ response line بإرجاع القيمة ٢٠٠ متبوعة بالكلمة OK هذه تشكل وتشير إلى نجاح الـ request ويكون الـ response يحتوي على

الصفحة المطلوبة والبيانات من السيرفر. ومثال آخر هو الشيفرة ٤٠٤ عندما تقوم بطلب صفحة ويفشل السيرفر في الحصول عليها.

٢- HTTP Header

الـ response header يعتبر مشابه request hader الذي ناقشناه في الأعلى. وتتقسم المعلومات التي فيه أيضاً إلى ثلاثة أنواع:

أ - عامة GENERAL: معلومات عن الـ client أو السيرفر ولاتخصص إلى واحد منهما.

ب - شخصية Entity: يحتوي على معلومات عن البيانات التي يتم إرسالها بين السيرفر والعميل.

جـ - الإجابة Response: يحتوي معلومات عن السيرفر الذي قام بإرسال الرد وكيفية تعامله ومعاجلته للرد (Response).

كما قلنا سابقاً، يتكون من عده سطور ويتم وضع سطر فارغ للإعلام عن انتهاء الهيدر.

مثال:

```
HTTP/1.1 200 OK -the satus line
 16:12:23 GMT -general header،Date: Mon; 1st Nov 1999
Server: Apache/1.3.12 (Unix) (SUSE/Linux) PHP/4.0.2 -the response
 12:08:03 GMT -Entity Header، 29 Oct 1999،Last-modified: Fri
```

السطر الأول ناقشناه والسطر الثاني مفهوم من غير شرح، السطر الثالث يقوم بتحديد البرنامج تبع السيرفر ونوعه ونظام التشغيل القائم عليه والسطر الأخير يقوم بتعريف آخر وقت تم فيه تعديل أو تجديد الصفحة.

ملاحظة: قد يحتوي الهيدر على أكثر من هذه المعلومات أو معلومات مختلفة وهذا يعتمد على نوع الشيء المطلوب من السيرفر.

٣ - Http Body

إذا تمت معالجة الطلب بنجاح، فإن الـHTTP response Body يحتوي على كود الـHTML ويقوم مستعرض الإنترنت بتفسيرها وتحويلها إلى الصفحة النهائية التي تراها.

أين سكريبت الـPHP من ذلك كله ؟

أصبح الآن لدينا مفهوم جيد عن طريقة إرسال المستعرض طلب صفحة من السيرفر وكيفية استجابة السيرفر لهذا الطلب.

تكلمنا عن أن سكريبت الـ php يتكون من ثلاثة أشياء: نص وكود php وكود html، لانسطيع وصف الـhtml بأنها لغة برمجة بشكل جيد ونستطيع أن نقول أن الـ php لغة سكريبتات Scripting Language لأنها تضيف قدرات html عليها مثل الجداول والفريمات بكود html بداخل كود الـphp هناك لغات تسمي لغات سكريبتات قد تكون متآلفة معها مثل الجافا سكريبت والفيجول بيسك سكريبت باستثناء أن الفرق بينها وبين الـphp هو أن الـphp لغة تعتمد على جهة المزود أي السيرفر ويمكنك تخصيص المتصفح الذي يستعرضها.
تجعلنا الـhtml نضمن سكريبتات الـphp فيها ضمن قواعد لذلك لكي نستطيع تشغيلها ولكننا لاننس أن امتداد الملفات يظل كما هو php أو php3 بدون تغير فيه لكي يتم إرسال السكريبت إلى مكتبة الترجمة (scriptingengine) التي تقوم بترجمة السكريبت إلى html (كأنك تترجم من عربي إلى إنجليزي أو العكس)

مفهوم الـ parsing و الـ Execution:

ممكن أن نقسم عملية الترجمة الذي يقوم بها سيرفر php إلى قسمين أو عمليتين:

العمليـة الأولى: هـي أن السيرفر يقـوم أولاً بفحـص قواعـد اللغـة وهـذا لايـضمن أن السكريبت صحيح مئة بالمئة ولكنه تدقيق في الأوامر وقواعد اللغة وهذا ما يسمونه بالـParsing

العمليـة الثانيـة: هـي تنفيـذ الـسكريبت بعـدها وإخراجـه علـى شـكل كـود html وهذا ما يسمي بالـ Execution.

بقي أن نقول أمراً معروفاً وهو أن السكريبتات نوعين:

١ - وهو ما ينفذ من جهة المزود

Server –Side scripting

٢ – ما ينفذ من جهة المستعرض (صفحة انترنت).

التعليقات

ما رأيك إذا كنت في شركة وكان معك أكثر من مبرمج وأردتم تصميم برنـامج، إذن قد تحتاجون لتنظيم العمل وتعديله لذا من اللازم أن تقوم بعمل توضيح لفائدة الكود الذي كتبته كي يسهل فهمه عليهم وإضافة تعديلات مناسبة، إذن التعليقات تستخدم في الإفادة عن شرح الأكواد أو إضافة معلومات لاتستعمل إلا كتوضيح أو أي شي آخر.

يمكنك عمل تعليق من سطر واحد كالتالي:

```
<?
//هذا تعليق لافائدة له وليس له أي معنى
?>
```

مثال آخر:

```
<?
//هذه الدالة تقوم بطباعة الكلمة تعليق
Echo "تعليق";
?>
```

وأيضاً يمكنك استخدام تعليق من أكثر من سطر كالتالي:

```
<?
/*تعليق يتكون من
أكثر من سطر بعلامة السلاش والنجمة
*/
?>
```

المتغيرات

ماهي المتغيرات ؟

أبسط تعريف يمكن أن نقوله عن المتغير هو أنه مساحة من الذاكرة تستخدم لتخزين المعلومات ويتم التحكم فيها عن طريق المبرمج في الـPHP ، المتغيرات تبدأ بعلامة الـ$ ولكي تقوم بإدخال قيمة في المتغير فإنك تستخدم المعامل (=) إذن لكي تقوم بإنشاء متغير يحتوي على قيمة يمكنك القيام بذلك كالتالي:

```
$alfares = "How Are You Every Body?";
$اسم_المتغير = قيمة;
```

لاحظ أن السطر السابق يتكون من خمسة أشياء:

١ / المتغير وهو alfares

٢ / وقبله علامة الـ$ لكي يعرف مترجم الـPHP أنه متغير

٣ / المعامل (=)

٤ / الفاصلة المنقوطة (;)

٥ / القيمة وهي How Are You Every Body? وهي القيمة الموجودة في المتغير أو التي اقترحناها للمتغير أو التي وضعناها فيه (لأن الذي اقترح القيمة هو أنت (مبرمج الphp))

ملاحظات:

١- أسماء المتغيرات حساسة لحالة الأحرف إذا كانت كبيرة وصغيرة

```
<?
 = “salem”; $Ahmed
$ahmed = “slmoon”;
echo $ahmed;
echo  $Ahmed;
?>
```

المتغيرين الذين بالأعلي مختلفين بسبب حالة الأحرف.

٢ - يمكنك استخدام المعامل (_)

$First_name

٣ - يمكنك استخدام ألف حرف في تسميه المتغيرات (وفي الواقع هي غير محددة).

علامات التنصيص

وهذه نقطة مهمة وهي لماذا وضعنا علامات التنصيص هذه ؟ فالإجابة تكون هي أن القيمة التي وضعناها حرفية ، أي تتكون من نصوص وهناك أنواع للمتغيرات وعلى ذلك سنفصل ونقول:

هناك أنواع للبيانات وهي:

١ - strings (حروف)

```
$Exa = “Just An Example”;
$Exa2 = “2.5”;
$Exa3 = “2”;
```

٢ - Integer (أرقام)

```
$Exam = 5;
```

٣ - Double (أرقام ذات فواصل)

```
$num= 5.4
```

٤ - array

يأتي تفصيلها فيما بعد

٥ - objects

تفصيلها في فصول أخرى

٦ - Unknown.

يأتي تفصيلها في فصل آخر.

المتغيرات لا يتم تعريف نوعها من قبل المبرمج إنما مترجم الـPHP يقوم بالتعرف عليها لكي يتم إتمام العمليات المختلفة عليها.

البيانات الحرفية /

في الـPHP أي قيمة تكون بين علامتي تنصيص عادية أو علامة تنصيص مفردة يعتبرها الـPHP قيمة حرفية

أمثلة :

"هذا النص بين علامتي تنصيص عادية أو مزدوجة"

'هذا النص بين علامتي تنصيص مفردة أو وحيدة'

يجب أن يبدأ النص وينتهي بنفس علامة التنصيص، وإلا فلن يتعرف الـPHP على القيمة الحرفية أو على النص.

```
<?
$d="غلط'
echo "خطأ '
?>
```

لايمكنك أيضاً أن تقوم بوضع علامة تنصيص من نفس النوع الذي تستخدمه القيمة الحرفية في وسط العبارة الحرفية أو النص.

```
<?
$variable = "هذا النص "خطأ بسبب وجود علامة في النص من نفس النوع";"
?>
```

وتصحيحه

```
<?
$variable = "'صحيح'هذا النص;"
?>
```

وأيضاً مثال آخر

```
<?
$r = “This is”BAD”; // خطأ
$t = “This is ‘good”; // صحيح
?>
```

أما إذا كنت مصراً على ذلك أو تحتاج إليها في عمليات ضرورية (كما سوف نرى فيما بعد حاجتنا إليها في صناعة النماذج) فيمكنك وضع معامل (\) قبل علامة التنصيص.

لكي تعمل معك بكل سهولة.

مثال:

```
<?
$u = “This Only An \” Example\” To Make You Understand Nothing”;
?>
```

حسناً ما رأيك لو أردنا أن نطبع المعامل (\) بنفسه؟

الحل هو أن نتبعه بمثله ، وبالمثال يتضح المقال:

```
$file = "c:\windows\system.ini";
echo $file;    // النتيجة c:windowssystem.ini

$file = "c:\\windows\\system.ini";
echo $file;    // النتيجة c:\windows\system.ini
```

يمكنك الجمع بين أكثر قيم المتغيرات في متغير واحد عن طريقة الـ(.)

```
<?
$first = "مؤسسة" ;
$last = "المطور العربي";
$fullname = $first. $last
Echo $fullname ;
//ولكننا نريد وضع فراغ بين الكلمتين
$fullname=  $first. ' '. $last   ;
Echo $fullname ;
?>
```

وأيضاً يمكننا أن نضيف إلى متغير قيمة متغير آخر:

```
<?
$f="I Love M" ;
$k= "y Country" ;
//إضافة القيمة إلى المتغير
$f  = $f. $k;
echo $f;
?>
```

```
<?
//تقريباً نفس العملية
 $f="I Love M" ;
$k= "y Country" ;
$f.=$k;
echo $f;
?>
```

الأرقام

العدد الفردي والمزدوج

الاختلاف المعروف لدي أنا حتى الآن هو أن الفرق بينهما هو الفاصلة العائمة (والله حتى إعطاؤها هذا الاسم يجعلنا نشعر بالإحباط والخوف)

لاحظ أننا لا نستخدم علامات التنصيص وذلك ليعرف الـPHP أنها بيانات رقمية قد نستخدمها في عمليات حسابية معقدة ويمكننا تطبيق عمليات حسابية بسيطة عليها إذا كانت حرفية.

```
//هذا عدد فردي
$j=2
//هذا عدد مزدوج
$h=4.5
```

العمليات الحسابية

هي مثل الجمع والطرح والضرب والقسمة وهي مرتبة كالتالي:

أولاً /الأقواس

ثانياً /الضرب ثم القسمة.

ثالثاً / الطرح ثم الجمع.

```
<?
Echo 5*2/5;
Echo 5*(2/5) ;
?>
```

مثال آخر:

```
<?
Echo 5-6+9 ;
?>
```

مثال لعملية حسابية نستخدم فيها متغير حرفي

```
<?
$W="2L";
$E= 2;
$F = $W * $E;
echo $W.' '.$E.' '.$F;
?>
```

مثال لعملية أخري لكنها لم تعمل وعليك استنباط السبب بنفسك:

```
<?
$W=”L10”;
$E= 2;
$F = $W * $E;
echo $W.’ ‘.$E.’ ‘.$F;
?>
```

يمكننا إضافة رقم واحد إلى متغير بثلاث طرق متنوعة:

مثال

```
$j++
```

أو

```
$j = $j+1
```

أو

```
$j += 1
```

ويمكننا على ذلك إضافة المتغير إلى نفسه كالتالي:

```
$j += $j
```

أو كالتالي:

```
$j = $j + $j
```

متغيرات النظام

هناك متغيرات يستخدمها النظام يمكنك أن تستعملها ومنها

$HTTP_USER_AGENT

التي تظهر لديك نوع المستعرض الذي يستخدمه العميل

مثال:

```
<?
Echo $HTTP_USER_AGENT ;
?>
```

الثوابت

يمكننا تعريف الثوابت بقول إنها قيم ثابتة لا تتغير ونعرفها عن طريق الدالة

define

الثوابت حساسة أيضاً لحالة الحروف

```
<?
 "alfarees");،Define ("author"
Echo "author is ". author ;
?>
```

هناك ثوابت يستخدمها النظام مثل:

PHP_OS

التي تقوم بعرض نظام التشغيل الذي يستخدمه السيرفر

مثال:

```
<?
Echo PHP_OS;
?>
```

معرفه وتحويل أنواع البيانات

إذا أردت أن تعرف نوع متغير ما يمكنك استخدام الدالة gettype

مثال:

```
<?
$n=5;
$l ="hi";
echo "The n Is ". gettype ($n). "<br>";
echo "The l is ". gettype ($l);
?>
```

إذا أردت تحويل نوع متغير ما يمكنك ذلك باستخدام الدالة settype:

مثال:

```
<?
$n = 10 ;
echo "Before is ". gettype ($n)."<br>";
"string");،settype ($n
echo "After That is go ". gettype ($n);
?>
```

الدالة isset

لمعرفة إذا كان المتغير منشأ مسبقاً أم لم يتم إنشاؤه وهي لا تتطلب غير اسم المتغير الذي تريد فحص وجوده

وتقوم بإرجاع القيمة (١) إذا كان المتغير تم إنشاؤه ولا ترجع أي قيمة إذا كان المتغير غير منشأ أو موجود.

مثال:

```
<?
$n = "n";
Echo isset ($n);
?>
```

الدالة unset

تقوم بحذف المتغير إذا كان موجوداً وتحرير الذاكرة منه (لذلك تأكد جيداً قبل استخدام هذه الدالة من إعطاء دمعة الوداع للمتغير المسكين)

```
<?
$n = "n";
unset ($n);
Echo isset ($n);
?>
```

الدالة empty

تقوم بإرجاع القيمة (١) إذا كان المتغير غير منشأ أو أن القيمة التي فيه صفر (٠) أو نص فارغ ("") ولا تقوم بإرجاع أي شيء إذا كان المتغير منشأ وفيه قيم غير المذكورة.

داوال الوقت والتاريخ

نستطيع إيجاد الوقت والتاريخ عن طريق دوال في الـPHP من تلك الدوال الدالة gmdate ()

مثال:

```
<?
Echo gmdate (m);
Echo gmdate (M);
?>
```

لاحظ أن هناك فرق في النتائج مع أننا نستخدم نفس الحرف لكن طريقة العرض تختلف عندما يكون الحرف كبيراً أو صغيراً.

تحتجز الـphp بكثير من الدوال والكلمات المحجوزة التي تقوم بعمليات مختلفة مثل العمليات الحسابية المعقدة والقيام بإيجاد الوقت والتاريخ وإرسال الرسائل البريدية وإيقاف السكريبتات لعدة ثواني هذه الدوال ليس مطلوب منك أن تحفظها كما تحفظ اسمك إنما المطلوب منك أن تفهم ماهية عملها واستخدامها في الوقت الذي تراه مناسباً.

يمكنك أيضاً عرض اليوم والشهر.

مثال:

```
<?
Echo gmdate ("M D");
?>
```

لاحظ أننا استخدمنا علامات التنصيص لكي تنجح العملية عندما قمنا باستخدام أكثر من عامل في الدالة

جرب استخدام الكود التالي:

```
<?
 d M Y H:i:s") ،Echo gmdate ("D
?>
```

النماذج

النماذج في الويب أو صفحات الإنترنت عبارة عن استمارات تقوم بتعبئتها ثم عند إرسالها لخادم الويب (السيرفر) يتلقاها برنامج يقوم بإجراء العمليات عليها مثل JavaScript أو ASP أو php (في حالتنا).

فائدة النماذج

لنقل إنك مثلاً أردت شراء كتاب من الإنترنت فإنك في الواقع تحتاج إلى تعبئة استمارة ببياناتك ورقم بطاقة الائتمان وغير ذلك من المعلومات ويتم ذلك عن طريق نموذج (فورم).

في الواقع أنت تقوم باختيار الكتاب الذي تريد وتكتب اسمك ورقم هاتفك وصندوق بريدك (ربما) في فراغات أو عن طريق الإشارة إلى مجموعة من الخيارات.
يتم تخزين هذه القيم في المتغيرات التي يتم كتابتها في الخاصيه name (نتكلم عنها في هذا الفصل) ويتم ارسالها عند ضغط زر - إرسال البيانات - (submit) إلى (البرنامج) الصفحة التي سوف تقوم بمعالجة هذه البيانات (والتي يتم تحديدها في الخاصية ACTION) وإجراء العمليات عليها مثل تخزينها مثلاً في قاعدة البيانات أو إرسالها إلى البريد الإلكتروني وذلك عن طريق الـphp.

ماذا يعمل العميل في النماذج ؟

إنه باختصار يقوم بتعبئة مربعات نصوص (textBox) ويقوم بوضع علامة صح في مربعات الاختيار(check boxes) أو يقوم بالتصويت أحياناً لشي معين فيختار زر اختيار (أزرار الراديو).
هذه الأشياء كلها يتم إنشاؤها بواسطة الـhtml وفصلنا يناقش كيفية إنشاؤها وكيفية التعامل والحصول على البيانات منها ، بقي علينا كبداية أن نعرف أن هذه الأدوات تنشأ في الواقع بين وسمين من وسوم لغة الـhtml وهي الوسمين
<form> و </form>

خصائص النماذج

يجمع النموذج جميع خصائص المضيف لكننا هنا سنتطرق إلى اثنين منهما وهما ACTION و METHOD التي تستخدم بكثرة و مهمة لنا في فصولنا القادمة

أما (ID;ClASS;NAME) فيلزمها تعمق في HTML خاصة عندما ندخل في ACCEPT-CHAR و ENCTYPE وستكون خارج نطاق موضوعنا حالياً وقد نفصلها في فصول قادمة إن شاء الله.

ACTION

وظيفة هذه الخاصية أن تخبر السيرفر مكان الصفحة التي يقوم بارسال معلومات النموذج إليها أوعنوانها أياً كان نوعها، وطبعاً في حالتنا ستكون الصفحة الثانية هي الصفحة التي تحتوي على سكريبت الـphp.

ليس مهماً أن تكون الصفحة php فقد تكون html ولكنها تحتوي على كود يختص بالتعامل مع برنامج تفاعلي لصفحات الويب مثل الجافا.

ولا نريد أن نخرج عن نطاق الموضوع فدعنا نعطي مثالاً على هذه الخاصية:

```
<FORM ACTION ="TEST.PHP">
.....
</FORM>
```

METHOD

هذه الخاصية تقوم بإخبار النموذج طريقة إرسال المعلومات إلى الصفحة الهدف وفي الواقع هناك طريقيتين مشهورتين ومعروفتين لإرسال المعلومات هما GET وPOST.

```
<FORM ACTION ="test.php" METHOD = "GET">
```

أو

```
<FORM ACTIN = "test.php" METHOD ="POST">
```

ملاحظه / في الواقع يوجد اكثر من هذه الطريقيتن لارسال المعلومات وهي (CONNECT;HEAD;OPTIONS:DELETE:TRACE) وغيرها ولكن لا تستخدم إلا بشكل نادر.

دعنا الآن نفصل هاتين الطريقتين بشكل أوسع:

GET

تقوم هذه الخاصية بإخبار مستعرض الإنترنت لديك بأن يقوم بإضافة المعلومات التي تمت كتابتها في النموذج إلى متصفح الإنترنت لديك وتكون طريقة كتابته كالتالي:

١- كتابه عنوان الصفحه المصدر.
٢- اتباعها بعلامة استفهام.
٣- كتابة العناوين والقيم.

http://localhost/test.html?name=value

قد تكون النقطيتن الأخيرتين غير مفهومتين بشكل جيد بسبب أنك لم تتعامل مع النماذج من قبل.

لكن الحقيقة أن النموذج يتكون من عناصر (مربع علامة، مربع نص، زر اختيار) ولكل من هذه العناصر عنوان خاص بها (name) ولكل منها قيمة خاصة بها (value).

وهي مشابهة للمتغيرات ويمكن أن يحتوي عنوان الصفحة على أكثر من عنوان (name) وأكثر من قيمة (value) ويقوم بالتعريف عنهما باستخدام المعامل (&).

مثال:

http://localhost/test.html?animal=cat&age=30

تسمى الإضافة التي تظهر بعد علامة الاستفهام (query String) نتيجة الاستعلام الحرفية.

العنوان دائماً يكون باللغة الإنجليزية (name) ونعامله كأنه اسم متغير من المفترض تعريفه في الصفحة الهدف (التي سنكتبها بالـPHP).

قد تحتوي القيم على فراغات أو معاملات مثل (+،-،\،#،٪)

يقوم المتصفح باستخدام لغة تشفير الصفحات URL ENCODING.

أيضاً يستخدم الـURL ENCODING مع الأحرف العربية أو اللغات الأخرى غير الإنجليزية في كتابة الحرف.

URL Encoding

هناك بعض الأحرف لا يستطيع المتصفح إضافتها لعنوان الصفحة بصيغتها الحقيقية بل يستخدم لغة التشفير في التعريف عنها وهذه جداول بالرموز الذي يستخدم المتصفح كود بدلاً من عرضها بصيغتها الحقيقية

الحرف	شفرته	الحرف	شفرته	الحرف	شفرته
Tab	%09	(	%28	;	%3B
Space	%20	)	%29	<	%3C
!	%21	+	%2B	>	%3E
“	%22	،	%2C	=	%3D
#	%23	.	%2E	?	%3F
@	%40	/	%2F	%	%25
\	%5C	:	%3A	&	%26

لاتقلق فليس عليك أن تحفظ كل هذه العلامات وتشفيراتها بل سيقوم المتصفح بالعمليه كلها بدلاً عنك.

POST

في الواقع وظيفتها هي نفس وظيفة الـ get ولكنها لا ترسل المعلومات في عنوان صفحة الإنترنت بل تقوم وضعها في الـbody التابع لـhttp response.
بالإضافة إلى أنه يستطيع إرسال البيانات بكمية أكبر من الـGET.

أيهما تستخدم GET أم POST ؟

قد يكون العيب في الخاصية GET عدم سرية المعلومات التي تقوم بكاتبتها ومن الممكن أن تظهر للشخص الذي يجلس إلى جوارك... خاصة عندما تريد الحفاظ على سرية معلوماتك.
أضف إلى ذلك أنها غير مفيدة في النصوص الكبيرة الحجم.
ولكنها مفيدة في أشياء كثيرة فمثلاً محركات البحث يجب أن تستخدم هذه الخاصية لكي يستطيع المستخدم أن يستخدم عنوان البحث ويحتفظ به لوقت آخر ولا يقوم من جديد بكتابة الكلمة التي يبحث عنها.

أيضاً الـPOST مفيدة في إخفاء المعلومات واحتواء كميات كبيرة من البيانات ولكن لايمكن الاحتفاظ بعنوان الصفحة.... مع ذلك فإنها أيضا ليست جيدة في الحماية بحيث أن أي هاكر خبير يمكنه الحصول على المعلومات إذا لم يكن لها تشفير معين في نقلها..لكن إذا أردت فعلاً أن تجعلها محمية فيجب عليك استخدام اتصال محمي إلى سيرفر محمي.... أو مايسمونه (SCURE CONNECTION TO SCURE SERVER).

أدوات التحكم في النماذج:

في الواقع إن أدوات التحكم عبارة عن مربعات النصوص العاديه (التي يدخل فيها المستخدم اسمه وعنوانه) وأزرار الراديو (والتي يقوم المستخدم فيها باختيار شيء معين (مثل الوجبة المفضلة لديه أو المشروب المفضل) ومربعات الاختيار (التي تتيح للمستخدم أن يختار ما يشتهي ويحب من الخيارات المعروضة)
وأيضاً القوائم التي تساعدك على اختيار أكثر من شيء أو شيء واحد.

في أغلب هذه الأشياء يتم استعمال الوسم
<INPUT>
وتلخيص تفصيله كالتالي:

```
<INPUT TYPE= type NAME= name VALUE= value other attribute>
```

الشرح:

١- TYPE= type

نحدد نوع الكائن إذا كان زر راديو أو مربع نص عادي أو مربعات الاختيار.

٢- NAME= name

تقوم فيها بإعطاء اسم لمتغير يتم حفظ القيمة فيه.

٣- VALUE= value

ستتضح وظيفته أكثر عندما ندرج عليه أمثلة إذ إن عمله يختلف من أداة إلى أخرى.

تطبيقات عملية

سنقوم في هذه التطبيقات بصنع برامج بسيطة تتكون من ملفين، الملف الأول يحتوي على كود HTML يقوم بتكوين النموذج والملف الثاني يقوم باستقبال النتائج وطباعتها.

مربعات النصوص [TEXT Box]:

نقوم بعمل ذلك كالتالي:

١ - قم بتشغيل محرر النصوص لديك.

٢ - اكتب الكود التالي:

```
<html dir ="rtl">
<FORM METHOD  = "GET" ACTION = "textbox.php">
ماهي وجبتك المفضلة في الصباح ؟


<br>
<INPUT TYPE ="text" NAME = "food" value="جبنة ومربى">
<br>
<INPUT TYPE= submit  VALUE="إرسال">
<INPUT TYPE= reset  VALUE="مسح">
</form>
</html>
```

٣ - قم بحفظ الملف كصفحة HTML. وقم بتسميته (textbox.html).

٤ - افتح محرر النصوص إذا كنت أغلقته.

٥ - اكتب الكود التالي:

```
<?
Echo " وجبتك المفضله والتي تموت في حبها هي  ". " ". $food ;
?>
```

٦ - قم بحفظ الملف كـ php. وقم بتسميته textbox.php.

٧ - الآن قم بأخذ الملفين وضعهما في مجلد السيرفر لديك.

٨-قم بتشغيل السيرفر واكتب في مستعرض الإنترنت لديك

http://localhost/textbox.html

٩ - قم بكتابة وجبتك المفضلة واضغط زر إرسال.

١٠- ستظهر النتيجة.

لاحظ كيف ظهر العنوان:

http://localhost/textbox.php?food=%CC%C8%E4%C9+%E6%E3%D1%C8%ED

الشرح

لقد قمنا في البداية بعمل صفحة تتكون من نص و مربع نص وزر يقوم بعملية إرسال البيانات

قمنا بصناعة بداية النموذج بواسطة الوسم <FORM> وقمنا بتحديد المكان الذي سيتم إرسال البيانات إليه بواسطة ACTION="textbox.php"

وقمنا بصنع مربع النص بواسطة الوسم INPUT واخترنا الـ TYPE="text"

كما قمنا بوضع القيمة الافتراضية فيه بواسطة القيمة Value= "جبنه ومربى"

وقمنا بوضع الناتج الذي يضعه المستخدم في مربع النص في المتغير food.

(لاحظ أن تسمية المتغيرات حساسة لحالة الأحرف في PHP وأننا لم نقم بوضع $ في صفحة المتغير في كود الـhtml).

وأيضاً لقد قمنا بإضافة زر بواسطة TYPE=SUBMIT

وقمنا بوضع كلمة على الزر وهي كلمة (إرسال) VALUE = "إرسال"

أيضاً قمنا بصنع زر آخر Type =reset

وقمنا بجعل العبارة التي عليه (مسح) Value="مسح"

هناك نوعين من الأزرار هي SUBMIT وRESET

١- الـ submit يقوم بإرسال المعلومات.

٢- الـ reset يقوم بمسح البيانات في جميع الأدوات في النموذج لإعادة إدخالها من جديد.

بعدما قمنا بإدخال البيانات وضغط زر الإرسال قام النموذج بإرسال البيانات إلى الصفحة المحددة في الخاصية ACTION وقامت الصفحة المحددة باستقبال النتائج الموجودة في النموذج وهي نتيجة واحدة في مربع نصوص تم حفظ قيمته في المتغير food.

وقامت بطباعتها بواسطة الدالة echo.

نظراً لأننا استخدمنا الأسلوب GET فقد تم إعطاؤنا عنوان الصفحة بالاضافة إلى (?) وأيضاً المعلومات المسجلة في المتغيرات والتي تم استخدام ال URL ENCODING فيها لأنها تستخدم حروف عربية.

مربعات النصوص الكبيرة (text area) طلبات أكبر للطعام الشهي !

إذا كنت تريد أن تكتب رسالة متعددة الأسطر فإنك تحتاج إلى أداة تحكم تختلف تماماً عن مربع النص العادي وهي مربعات النصوص الكبيرة التي يمكنك فيها من إدخال نصوص كبيرة الحجم ومتعددة الأسطر.

تستخدم هذه الأداة وسم فتح ووسم إغلاق

<TEXTAREA>

</TEXTAREA>

ويمكنك تحديد حجمها بواسطة تحديد الصفوف بالخاصية rows والأعمدة بالخاصية cols.

تمرين عملي

١- قم بفتح محرر النصوص لديك

٢- قم بكتابة الكود التالي:

```
<html dir=”rtl” >
 METHOD=”POST”> <FORM ACTION = “TAREA.PHP”
ما هي وجبتك المفضلة ؟
<br>
<TEXTAREA NAME = “food” ROWS=”10” COLS =”50” >
جبنة
مربى
معكرونة
بيف برغر
</TEXTAREA>
<br>
”>قم بإرسال الطلبات إلى النادل<INPUT TYPE = SUBMIT VALUE =”
</FORM>
</html>
```

٣- قم بحفظ الملف باسم TAREA.html.

٤- الآن قم بفتح ملف جديد في محرر النصوص.

٥- قم بكتابة الكود التالي:

```
<html dir=”rtl”>
وجبتك المفضلة هي:
<br>
<?
Echo $food;
?>
</html>
```

٦- قم بحفظ الملف باسم tarea.php

٧- قم بوضعهما في مجلد السيرفر لديك.

٨- قم بتشغيل البرنامج.

http://localhost/tarea.html

٩- قم بضغط الزر لإرسال البيانات.

١٠- شاهد النتيجة.

الشرح

لا نضيف شيئاً على قولنا هنا سوى أننا نريدك أن تلاحظ كيف جهزنا القيمة الافتراضية بكتابة نصوص بين وسومات الـtextarea وأيضاً أننا استخدمنا الأسلوب POST في إرسال البيانات مما جعلها لا تظهر في شريط العنوان.

وأن الـNAME تحدد اسم المتغير التي ستذهب إليه القيمة واسم المتغير في الكود لا يحتوى على $ لأنه كود HTML وليس PHP.

مربعات الاختيار (Check Box) أكثر من خيار في وقت واحد !

في الواقع قد نرى مربعات الاختيار في صفحات الويب عندما نريد الاشتراك في موقع معين لرؤية محتوياته أو عندما نريد تسجيل بريد إلكتروني أو حجز مساحة عند موقع.

وفائدتها هي إتاحة فرصة للمستخدم لتحديد أنواع الأشياء التي يريد أن يشترك فيها مثلاً أو إتاحة فرصة له لقبول إتفاقية أو غير ذلك أو رفض الجميع أو قبول الجميع.

يمكننا صنع مربع العلامة بواسطه الوسم INPUT

```
<INPUT TYPE = ”CHECKBOX” NAME = “swalif” value= ” محادثة ” checked>
```

نقوم بتحديد نوع الأداة بأنها مربع علامة في هذا الجزء

TYPE = ”CHECKBOX”

نقوم بتحديد اسم المتغير في هذا الجزء
NAME = “swalif”
ونقوم بتحديد القيمة التي يتم وضعها في المتغير إذا قام المستخدم باختيار مربع العلامة في هذا الجزء:
" =valueمحادثة "
إذا لم تقم بوضع الخيار value فستكون القيمه الافتراضية هي on عند اختيار المستخدم مربع العلامة وستكون فراغ إذا لم يقم المستخدم باختيار المربع.

ونقوم بوضع القيمة الافتراضية بإضافة الكلمة checked فإذا تم وضع هذه الكلمة يكون مربع العلامة مختار تلقائياً أما إذا لم نتكبها فسيكون بدون علامة الاختيار.
Checked

تطبيق عملي (١):

١- قم بفتح المفكرة وقم بكتابة الكود التالي:

```
<html dir=”rtl”>
<FORM ACTION=”CHECK.PHP” METHOD = “POST”>
ما الذي تريد أن تفعله في الحياة ؟ (يمكنك اختيار أكثر من إجابة)
<br>
<INPUT TYPE=”CHECKBOX” NAME = “WIFE” CHECKED>
الذي أريد أن أفعله في الحياة هو أن أعمل.
<br>
<input type= submit value = “إرسال”>
</FORM>
</html>
```

٢- قم بحفظ الملف باسم check.html.

٣- قم بفتح ملف جديد في المفكرة وقم بكتابة التالي:

```
<?
Echo $WIFE ;
?>
```

٤- قم بحفظ الملف باسم check.php.

٥- قم بنقل الملفين إلى مجلد السيرفر.

٦- اكتب في المتصفح

http://localhost/check.html

٧- النتيجة.

تطبيق عملي (٢):

١- افتح المفكرة واكتب الكود التالي وقم بحفظه في ملف جديد باسم check2.html

```
<html dir=”rtl”>
<FORM ACTION=”CHECK2.PHP” METHOD = “POST”>
ما الذي تريد أن تفعله في الحياة ؟ (يمكنك اختيار أكثر من إجابة)
<br>
<INPUT TYPE=”CHECKBOX” NAME = “WIFE” value= “عمل” CHECKED>
الذي أريد أن أفعله في الحياة العمل.
<br>
<INPUT TYPE=”CHECKBOX” NAME = “jihad” value= “أتعلم” >
أن أتعلم أكثر
<br>
<INPUT TYPE=”CHECKBOX” NAME = “qran” value= “أولادي” CHECKED>
أن أربي أولادي بشكل صالح
<br>
<input type= submit value = “إرسال”>
</FORM>
</html>
```

٢- قم بفتح ملف جديد وقم بوضع الكود التالي فيه:

```
<html dir = “rtl”>
<?
Echo $WIFE. “ “. $jihad. “ “. $qran ;
?>
</html>
```

٣- قم بحفظه باسم check2.php

٤- قم بتشغيل الملف.

٥- النتيجة.

تطبيق عملي (٣)

١- افتح محرر النصوص واكتب الكود التالي:

```
<html dir=”rtl”>
<FORM ACTION=”CHECK3.PHP” METHOD = “POST”>
ما الذي تريد أن تفعله في الحياة ؟ (يمكنك اختيار أكثر من إجابة)
<br>
<INPUT TYPE=”CHECKBOX” NAME = “alswalif[]” value= “عمل” CHECKED>
الذي أريد أن أفعله في الحياة هو أن أعمل
<br>
<INPUT TYPE=”CHECKBOX” NAME = “alswalif[]” value= “أتعلم” >
أن أتعلم أكثر
<br>
<INPUT TYPE=”CHECKBOX” NAME = “alswalif[]” value= “أولادي” CHECKED>
أن أربي أولادي بشكل صالح
<br>
<input type= submit value = “إرسال”>
</FORM>
</html>
```

٢- قم بحفظه باسم check3.html وافتح محرر النصوص من جديد واكتب الكود التالي:

```
<html dir=”rtl”>
<?
Echo “$alswalif[0] <br>” ;
Echo “$alswalif[1] <br>” ;
```

```
Echo  "$alswalif[2] <br>" ;
?>
</html>
```

٣- قم بحفظه باسم check3.php وقم بنقلهما إلى ملف السيرفر.

٤- قم بتشغيل البرنامج

http://localhost/check.html

٥- قم بضغط زر إرسال وانظر للنتيجة

الشرح

في الواقع لقد قمنا بتطبيق ثلاث تمارين **التمرين الأول** أردنا لفت النظر إلى أننا قمنا بعدم استخدام value للمتغير وتم إعطاء القيمة on عند اختيار المستخدم مربع العلامة بالإضافة أن مربع العلامة كان مختاراً بسبب وضعنا الخاصية CHECKED ولكن التمرين غير عملي وغير جيد بدون وضع قيم VALUE عند وضعنا لأكثر من مربع اختيار لذلك فقد قمنا بإضافة قيم يتم وضعها في المتغيرات عند اختيار المستخدم لها كما في **التمرين الثاني** وأردنا لفت النظر في التمرين إلى شيء يسمي بالمصفوفات فإذا أردنا مثلاً أن نجعل اسم المتغير متشابهاً وإجراء عمليات تكون أسرع عليه نستخدم المصفوفات ولن نتطرق إلى المصفوفات حالياً ولكن أردنا لفت نظرك فقط وسنقوم بالتكلم عن المصفوفات بالتفصيل لاحقاً هي والتكرارات بعد التكلم عن العبارات الشرطية في الـPHP.

أزرار الراديو [RADIO BUTTONS] [اختر المشروب المفضل !]

ماهو اختيارك المفضل ؟ علماً بأنه لا يمكنك اختيار أكثر من خيار واحد !!

في الواقع إن زر الراديو يتيح لك أن تختار شيء واحد من بين عدة اختيارات ونراه كثيراً عند اتفاقيات البرامج حيث يعطيك فرصة إما بقبول الاتفاقية أو رفضها ويكون واحد من الاختيارين محدداً (وهو خيار الرفض!).

يتم استخدام أزرار الراديو باستخدام العبارة <INPUT> كالتالي:

```
<INPUT TYPE = ”radio” NAME = “name” value= ”value” checked>
```

نقوم بتحديد نوع الكائن بأنه زر راديو في هذا الجزء:

TYPE = ”radio”

نقوم بتحديد اسم المتغير في هذا الجزء:

NAME = “name”

نقوم بتحديد القيمة التي ستكون في المتغير هنا:

value= ”value”

في الواقع مع أزرار الراديو نقوم بجعل اسم المتغير name هو نفسه والقيم مختلفة value
لكل سؤال. وإذا لم نقم بوضع قيمة فسيقوم PHP بوضع القيمة on للمتغير.

تطبيق عملي:

١- قم بتشغيل محرر النصوص لديك واكتب الكود التالي وقم بحفظه في ملف اسمه radio.html.

```
<html dir=”rtl”>
<form action = radio.php method = “post”>
ما هو مشروبك المفضل ؟
<br>
<br>
<INPUT TYPE = ”radio” NAME = “mshroob” value= ”شاي” checked>
شاي
<br>
<INPUT TYPE = ”radio” NAME = “mshroob” value= ”قهوة” >
قهوة
<br>
<INPUT TYPE = submit  value= ”إرسال” >

</form>
</html>
```

٢- قم بفتح محرر النصوص واكتب الكود التالي وقم بحفظه باسم radio.php

```
<html dir = "rtl">
<?
echo "مشروبك المفضل هو:". " ". $mshroob;
?>
</html>
```

٣ - قم باختيار المشروب المفضل واختر إرسال.

الشرح:

في الواقع لقد قمنا بصنع أزرار راديو ولقد قمنا بوضع قيمة لكل زر تكون تابعة للعباره التي بجوار الزر. ولقد قمنا بوضع عبارة checked لكي ترى كيف أن الأداة التي تحتوي على العبارة تكون محددة تلقائياً ولاحظ أن العبارة التي تكون بجانب الزر تكون موجودة أسفل كود الزر مثل:

```
<INPUT TYPE = ”radio” NAME = “mshroob” value= ” شاي” checked>شاي
```

وأيضاً لاحظ أننا استخدمنا متغيراً واحداً فقط لجميع الاختيارات بحيث أن جميع الأزرار قيمتها تعود إلى هذا المتغير.

القوائم [Lists Or drop down menus] اختر مواصفات زوجتك للمستقبل واسمها:

تستخدم القوائم في الـ html بشكل مختلف قليلاً عن الأداوت السابقة إذ إننا نستخدم وسمين من وسوم لغة html وهما: <select> لنقوم بإنشاء القائمة و <OPTION> ونستخدم الخاصية MULTIPLE إذا كنا نريد إتاحه الفرصة للمستخدم أن يختار أكثر من قيمة ونقوم بوضع القيمة التي يختارها المستخدم في متغير بواسطة الخاصية NAME أو في مصفوفة متغيرات (وسيتضح مفهوم المصفوفات لديك جيداً في فصل المصفوفات بإذن الله).

تطبيق عملي:

١- قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه في ملف باسم lists.html:

```
<html dir=”rtl”>
<form action = “lists.php” method = “post”>
ماذا تريد أن يكون اسم زوجة المستقبل(لغير المتزوجين) ؟
<br>
<select name = “wife” >
<option>هناء</option>
<option>جمانة</option>
<option>رزان</option>
<option>سحر</option>
<option>سارة</option>
<option>سمية</option>
```

```
</option>روان<option>
</option>دلال<option>
</option>اسم اخر<option>
</select>
<BR>
ماذا تريد أن تكون مواصفاتها ؟
<Br>
<select name="dis[]" multiple>
</option>جميلة<option>
</option>متدينة<option>
</option>شقراء<option>
</option>جعداء الشعر<option>
</option>سوداء<option>
</option>سمراء<option>
</option>بيضاء<option>
</select>
<br>
">إرسال<INPUT TYPE=SUBMIT VALUE="
</html>
```

٢- قم بفتح ملف جديد واكتب فيه الكود التالي وقم بحفظه باسم lists.php:

```
<html dir="rtl">
<?
 ". " ". $wife ;لقد أردت أن يكون اسم زوجتك Echo "
Echo "<br><br>";
 ";ولقد أردت أن تكون مواصفاتها Echo "

Echo "<br><br>";
Echo "$dis[0] <br>";
Echo "$dis[1] <br>";
Echo "$dis[2] <br>";
Echo "$dis[3] <br>";
Echo "$dis[4] <br>" ;
Echo "$dis[5] <br>";
Echo "$dis[6] <br>";
?>
</html>
```

قم بتشغيل البرنامج

http://localhost/lists.html

واختر ما تريد ثم اضغط زر إرسال

الشرح:

لقد قمنا بصناعة قائمة تسمح باختيار قيمة واحدة منها ثم تذهب هذه القيمة إلى المتغير wife وصنعنا قائمة ثانية تسمح باختيار أكثر من عنصر واحد وقمنا بوضع هذه القيم في مصفوفة متغيرات (سيتضح معني المصفوفات في فصول قادمة إن شاء الله).

الأداة الخفية [والمعلومات السرية!] [hidden control]

هناك بعض الأوقات تحتاج فيها إلى إرسال بعض المعلومات من صفحة ويب إلى صفحة ويب أخرى عن طريق النماذج وفي نفس الوقت أنت لا تريد المستخدم أن يقوم برؤية هذه المعلومات.

في الواقع هناك أداة تساعدك على إخفاء هذه المعلومات عن المستخدم يسمونها بحقل النموذج المخفي أو الأداة الخفية (hidden form field or hidden control).

هذه الأداة تلعب دوراً مختلفاً ومتميزاً عن بقية الأدوات وهي إخفاء المعلومات التي تم إدخالها كما شرحنا في السابق وهي مفيدة جداً مع النماذج المصنوعة بواسطة الـPHP إذ إنها تسمح لنا أيضاً بأن تكون المعلومات المخفية هي متغيرات PHP.

يتم صنع هذه الحقول المخفية كالتالي:

```
<INPUT TYPE=HIDDEN NAME =hidden1 VALUE="الرسالة السرية">
```

نقوم بوضع HIDDEN لكي يعرف المتصفح أن هذه المعلومات خفية (لا تظهر للمستخدم) ونضع اسماً للمتغير الذي يقوم بالاحتفاظ بالمعلومات والذي يتخزن اسمه في الـNAME ونقوم بوضع المعلومات التي نريد إخفاءها في الـVALUE.

نستطيع الاستفاده أيضاً منها عن طريق الـphp وذلك عن طريق كتابة كود الـHTML بواسطة الأمر echo() في الـPHP كما في المثال التالي:

```
<?
$msg1= " هذه العبارة لن تظهر";
Echo "<form>" ;
Echo "<input type=hidden name =secret value= '$msg1'>";
Echo "<input type=submit>";
Echo "</form>";
?>
```

هذا الكود الذي تراه عبارة عن كود HTML تمت كتابته بالـPHP عن طريق الأمر echo() ولقد استطعنا تخزين قيمة متغير php ($msg) في متغير html (secret).

تطبيق عملي:

١ - افتح محرر النصوص واكتب الكود التالي واحفظه باسم hid.php:

```
<html dir="rtl">
<head></head>
<body>
<?
$car1= " لكزس";
$car2= " ماكسيما";
$car3="لاندكروزر";
Echo "<form method =get action='hid2.php'>";
Echo " ماهي السيارة التي تتمنى أن تشتريها أو تحظى بها ؟";
Echo "
<select name= 'favcar'>
<option>$car1</option>
<option>$car2</option>
<option>$car3</option>
```

```
</select><br><br>
<input type =hidden name = hid1 value='$car1'>
<input type =hidden name = hid2 value='$car2'>
<input type =hidden name = hid3 value='$car3'>
<input type = submit  value='إرسال'>
</form>";
?>
</body>
</html>
```

٣- افتح محرر النصوص واكتب الكود التالي واحفظه باسم hid2.php

```
<html dir="rtl">
<head></head>
<body>
<?
Echo "لقد قمنا بعرض السيارات التالية عليك:<br>";
Echo "$hid1<br>";
Echo "$hid2<br>";
Echo "$hid3<br>";
Echo "<br>ولقد قمت باختيار:<br>";
Echo $favcar;
?>
</body>
</html>
```

٣- قم بنقل الملفين إلى مجلد السيرفر ثم قم بتشغيل السكريبت:

http://localhost/hid.php

الشرح:

لقد قمنا بعمل نموذج بسكريبت الـphp لاحظ أننا استخدمنا الـ(') بدلاً من (") كما كنا نعمل في الـ html وذلك لأننا قلنا سابقاً أن القيم الحرفية .... (راجع فصل المتغيرات) ولقد قمنا بإدراج قيم متغيرات الـphp في كود الـhtml مما يوفر علينا الكثير من إعاده الكتابة (في حال كان النص المستخدم طويلاً).

اقرأ المثال أكثر من مرَّ وسيتضح لك المقال أكثر باذن الله.

استخدام حقل كلمات السر [Password fields]

لكي تجعل المعلومات أكثر حماية من التعرض إلى السرقة أو غير ذلك يمكنك استخدام حقول كلمات السر الذي هو عبارة عن مربع نص بسيط يقوم بإظهار النص على شكل نجوم ❖❖❖❖ في حال كان الجهاز يستخدم على يد أكثر من شخص فإن هذه الطريقة جيدة قليلاً في أن لا يرى أي شخص معلومات الآخر السرية. في الواقع مع ذلك فإنك لاتكون قد أديت حماية إذا كان الأسلوب المستخدم في إرسال بيانات المستخدم هو الأسلوب get إلا إذا كنت تستخدم تشفير البيانات ويكون أكثر جودة إذا استخدمت الأسلوب post وأيضاً لن يكون محمياً من الهاكر إذا لم تكن تستخدم SSL (Secure Socket Layer) لكي تقوم بتنشيط تشفير البيانات.

تطبيق عملي

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم pass.php

```
<html dir="rtl">
<body>
<form method=post action="pass1.php">
اسم المستخدم
<br>
<input type="text" name = "user">
<br>
كلمة المرور
<input type="password" name = "pass">
<br>
<input type = submit  value='إرسال'>
</form>
<body>
</html>
```

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم pass1.php

```
<?
Echo "اسم المستخدم هو: ";
Echo "<br>$user<br>";
```

```
Echo "وكلمه المرور هي:";
Echo "<br><br>$pass"
<?
```

قم بنقل الملفين إلى مجلد السيرفر لديك.

قم بتشغيل البرنامج ولاحظ النتيجة.

إرسال البريد الإلكتروني بواسطة الphp:

البريد الإلكتروني هو الحياة التي تنبض بها السكريبتات فمثلاً هناك سكريبتات ارسال بريد إلى صاحب الموقع تخبره بشي معين أو ملحوظة أو غير ذلك ويمكن استخدامها في أكثر من مجال.

والدالة التي تستخدم في ذلك هي الدالة mail()

```
mail("$to","$sub", "$msg", "From:$you");
```

وتقوم بوضع بريد الذي ستصله الرسالة في الخانة $to وموضوع الرسالة في الخانة $sub والرسالة في الخانة $msg وبريدك أنت أو بريد المرسل في الخانة $you.

تطبيق عملي

قم بكتابة الكود التالي واحفظه في ملف باسم mail.html

```
<html dir=rtl>
<head>
<title>    برنامج إرسال بريد</title>
</head>
<body>
<form action="mail.php" method="post">
عنوان المرسل
<br>
<input type="text" name = "you">
```

```
<br>
عنوان المستقبل
<br>
<input type="text" name = "to">
<br>
موضوع الرسالة
<input type="text" name = "sub">
<br>
الرسالة
<textarea rows=10 cols=20 name = "msg" >
</textarea>
<input type="submit" value = "إرسال البريد الإلكتروني">
</form>
</body>
</html>
```

قم بإنشاء ملف آخر وقم بكتابة الكود التالي وقم بإعطائه الاسم mail.php.

```
<?
mail("$to"، "$sub"، "$msg"، "From:$you");
?>
```

قم بوضع الملفين في مجلد السيرفر وقم بتشغيل البرنامج واملا البيانات واضغط زر الإرسال وسترى ان الرسالة تم إرسالها بنجاح.

برامج عملية

برنامج أو سكريبت إرسال بطاقات بسيط يحتوي على ملفين الملف الأول فيه البطاقات وعنوان البريد الإلكتروني والملف الثاني هو الذي يقوم بعملية الإرسال

الملف الأول هو chcard.php وكوده كالتالي:

```
<html dir=”rtl”>
<form action =card.php method = “post”>
اختر البطاقة التي تريد إرسالها
        <br>
<br>
<INPUT TYPE = ”radio” NAME = “card” value= ”
http://www.khalaad.f2s.com/MADINA9_small.JPG” checked>
البطاقة الأولى
<br>
<br>
<img src="http://www.khalaad.f2s.com/MADINA9_small.JPG"
width="100" height="100" alt="" border=0>
<br>
<br>
<INPUT TYPE = ”radio” NAME = “card” value= ”
http://www.khalaad.f2s.com/Haram3.jpg” >
البطاقة الثانية
<br>
<img src="http://www.khalaad.f2s.com/Haram3.jpg" width="100"
height="100" alt="" border=0>
<br>
اسمك
<br>
<input type="text" name = "myname">
<br>
بريدك الإلكتروني
<br>
<input type="text" name = "you">
<br>
بريد صديقك
<br>
<input type="text" name = "to">
```

```
<br>
موضوع التهنئة
<input type="text" name = "sub">
<br>
الرسالة
<br><br>
<textarea rows=10 cols=20 name = "msg" >
</textarea>
<br>
" >إرسال البطاقة<INPUT TYPE = submit  value= "
</form>
</html>
```

الملف الثاني يقوم بعملية إرسال البطاقة وقم بكتابة الكود التالي وحفظه في ملف باسم card.php

```
<?
$message = "لقد قام $myname بإرسال بطاقة إليك " . "\n". "وهو يقول في نص رسالته
لك:".\n$msg" وتجدها على الرابط التالي: ."\n". "$card"."\n";
mail("$to"، "$sub"، "$message"، "From:$you");
echo "<center>مبارك، ، لقد تم إرسال الرسالة بنجاح </center>";
?>
```

ملاحظة:

الدالة \n تقوم فقط ببدء سطر جديد لأننا لا نستطيع استخدام
 في نص الرسالة.

الفصل الثالث
الأوامر الشرطية

لقد أخذنا في الفصول السابقة فكرة عن المتغيرات وكيفية تعامل البيانات مع النماذج..في هذا الفصل سنتعلم كيفية التحكم بالكود بمعنى تنفيذ سطر معين من الكود عند حصول شرط معين وعند عدم حصوله نتجاهل السطر ونتجه إلى السطر الذي يليه.. هذا يمنحنا تحكماً أكبر بالكود ويجعلنا نستخدم قرارات وتنفيذ أشياء ممتازة وبرامج رائعة بالـPHP.

دعنا نعطيك فكرة من حياتنا اليومية....

تقوم في الصباح وتريد أن تحضر فطورك الذي يتكون من التالي:

عسل

جبنة

خبز

شاي

ستقوم بالذهاب إلى الثلاجة ثم تقوم بالبحث عن الأشياء التي يتكون منها فطورك، فإذا لم تجد ما تريد تستعد للذهاب إلى المركز التجاري لشرائه، تذهب إلى المطبخ وتتأكد مرة أخرى وتبحث عن المؤونة التي يحتاجها البيت بشكل عام.

١- تبحث عن جبنة وإذا لم تجدها تنتقل إلى الخطوة ٣.

٢- إذا وجدت جبنة فإنك تبحث عن العسل فإذا وجدته تنتقل إلى الخطوة ٤، وإذا لم تجده تنتقل إلى الخطوة ٥.

٣- تقوم بكتابتها في ورقة جانبية وتقوم بالبحث عن العسل.

٤- تتجهز للذهاب إلى المركز التجاري.

٥- تكتبه في ورقة جانبية ثم تتجهز للذهاب إلى المركز التجاري.

هل لاحظت أنك كنت تقوم بالبحث عن أشياء معينة فإذا وجدتها (true) قمت بالبحث عن التي تليها وإذا لم تجدها (false) تقوم بتسجيلها في قائمة المشتريات لديك.

القيم المنطقية والدوال الشرطية

في الواقع لقد تكلمنا عن المتغيرات سابقاً وذكرنا بأن هناك متغيرات منطقية (قيمتها إما صحيح إم خطأ) ولم نقم بشرحها ، وهذا الفصل سيتولى شرحها وإعطاء أمثلة على كيفية التعامل معها.

العبارة IF

IF condition is true(إذا كان الشرط صحيحاً)
excute this code(قم بتنفيذ هذا الكود)

إن الدالة IF معروفة تقريباً في جميع لغات البرمجة... حيث أنها تقوم بعملية التحقق من شي معين وتنفيذ بعض الأشياء إذا كان الشرط صحيحاً (true) والقيام بتنفيذ أشياء أخرى إذا لم يكن صحيحا....

سيقوم الـPHP بتنفيذ الكود التي بين } و { فقط إذا كان الشرط صحيحاً.

أما إذا لم يكن صحيحاً فسيقوم بتجاوزه وتنفيذ الكود الذي يليه.

ويمكنك أيضاً أن تقوم بجعلها بسطر واحد ولا تستخدم الأقواس بل تكتب الأمر مباشرة:

```
IF condition is true excute function;
```

لاحظ أنه لا بد من استخدام } و { إذا كان الكود يتكون من عدة أسطر أما إذا كان يتكون من سطر واحد فلا داعي لاستخدامها.

فالمثالين التاليين كلهما صحيحين.

مثال(١)

```
<?
$S=10
IF ($S=10) echo 11;
?>
```

مثال (٢)

```
<?
$S=10
IF ($S=10){
echo 11;
}
?>
```

لنتخيل مثلاً أن الجو ممطر وسنقوم بإعطاء المطر متغيراً ونسميه rain ونقوم بإعطاء المظلة اسم متغير آخر ونسميه umbrella وسنقوم بافتراض أن هناك أمر في الـphp يسمي go out حسناً الآن الكود الذي نريد أن نقوم بكتابته هو:

```
If $rain = true
{
$umberrlla = true
}
go out();
```

فائدة هذا الكود هو أن تأمر الـPHP بحمل المظلة ($umberrlla=true) معه إذا كان الجو ممطراً ($rain=true) وإذا لم يكن ممطراً ولم يتحقق الشرط فإنه سيخرج إلى النزهة بدون أي مظلة.

طبعاً ليس هناك دالة تقوم بذلك إنما قمنا بذلك من أجل التوضيح للمستخدم هيكلية عمل الدالة بشكل عام.

مقدمة إلى القيم المنطقيه [Boolean Values]

القيم المنطقية ترمز إلى الأشياء التي لا تحتمل أكثر من احتمالين وهما إما صح وإما خطأ ، وهي نوع جديد من القيم غير التي كنت تعرفها سابقاً (مثل الرقمية والنصية).

مثال

```
<?
$variable=true;
echo "$variable";
?>
```

لو قمت برؤيه النتييجة ستجد أنه يطبع الرقم واحد وهو قيمة المتغير إذا كان صحيحاً ، أما إذا كان خطأ أو غير صحيح فقيمته ستكون(٠).

المعاملات المنطقية

لقد أخذنا المعاملات الرياضية فيما سبق بشيء من التفصيل (+،-،/،❖) والآن سنأخذ شيئاً جديداً من المعاملات وهي المعاملات المنطقية التي تساعدنا في صناعة الشروط والتقييدات على شيء معين وتعطينا تحكماً أكبر في الكود.

المعاملات: > و <

من المفترض أن تكون متآلفاً مع علامتي (الأكبر من) و (الأصغر من) في الرياضيات التي تتعلمها في المدرسة مما يجعل فهم هذا الأمثلة بسيطاً.

```
<?
If (6>5)
{
echo "الرقم ستة أكبر من الرقم خمسة ";
}
Echo "end";
?>
```

سيقوم الـPHP في مثالنا هذا بفحص الشرط (6>5) فإذا كان صحيحاً (true) سيقوم بطباعة السطر (الرقم ستة أكبر من الرقم خمسة) ثم يقوم بطباعة end، وإذا لم يكن صحيحاً فسيقوم بتجاهل الكود وطباعة (end) فقط.

يمكننا أيضاً استعمالها في المقارنة بين متغير ورقم أو بين متغير وثابت (constant) أو العكس أو المقارنة بين متغيرين.

مثال (١)

```
<html dir ="rtl">
<?
$LuckeyNumber = 5;
If ($LuckeyNumber<6)
{
echo ("رقم الحظ أصغر من الرقم ستة ");
}
?>
```

مثال (٢)

```
<html dir="rtl">
<?
$f=5;
$r=10;
If ($f >$r)
{
echo "المتغير $f أكبر من المتغير $r";
        }
?>
```

تطبيق عملي:

قم بتشغيل محرر النصوص واكتب الكود التالي واحفظه باسم thegame.php

```
<html dir = "rtl">
<body>
<form method =get action="game.php">
ماهو الرقم الذي أفكر به الآن والذي هو بين ١ و١٠ ؟
<input type="text" name="number">
<br>
<br>
<input type =submit>
</form>
</body>
</html>
```

قم بفتح محرر النصوص لديك من جديد واكتب الكود التالي واحفظه باسم game.php

```
<html dir="rtl">
<body>
<?
10);،$num = rand (1
if ($number>$num)
{
" ; لقد اخترت رقم أكبر من الذي أفكر فيه echo "
"; الرقم الذي أفكر فيه هو Echo "
Echo $num;
" ; يؤسفنا فعلاً أنك لم تنجح، نتمنى أن نقول لك في المرات القادمة Echo "<br>"."
}
if ($number<$num)
{
" ; لقد اخترت رقم أصغر من الذي أفكر فيه echo "
$num"; الرقم الذي كان في مخيلتي هو Echo "
"; يؤسفنا فعلاً أنك لم تنجح، نتمنى أن نقول لك في المرات القادمة Echo "<br>". "
}
?>
لقد نجحت
</body>
</html>
```

شرح التطبيق:

الداله rand

تقوم هذه الدالة باختيار رقم عشوائي من بين رقمين يتم اعطاؤها إياهما، الرقم الاول(X) هو الأصغر والرقم الثاني هو الأكبر(y)

```
y);،Rand (x
```

يمكنك حفظ القيمة التي تقوم بإخراجها هذه الدالة في متغير مباشرة

مثال

```
$Num = rand (5.57);
```

وهذا يوضح ماقمنا به في الكود

```
$num=rand(1، 10);
```

لقد قمنا باختيار قيمة عشوائية ثم قمنا بمقارنتها مع القيمة التي تم إدخالها من قبل المستخدم فإذا كانت القيمة التي أدخلها المستخدم أكبر من قيمة العدد العشوائي أخبرناه بأن الرقم الذي أدخله أكبر من الرقم الصحيح... وهذا ما تجده جلياً في الأسطر التالية:

```
if ($number>$num)
{
echo "لقد اخترت رقم أكبر من الذي أفكر فيه " ;
Echo "الرقم الذي أفكر فيه هو ";
Echo $num;
Echo "<br>"."يؤسفنا فعلاً أنك لم تنجح، نتمنى أن نقول لك في المرات القادمة " ;
}
```

فإذا لم ينطبق الشرط وكان الرقم الذي اختاره المستخدم أصغر من الرقم العشوائي فإنه يترك الشرط الأول ويتجه إلى الشرط الثاني ويطبق الأوامر التي فيه والتي تقوم بإخباره بأن الرقم الذي قام باختياره أصغر من الرقم المطلوب، وهذا ما تجده جلياً في الأسطر التالية:

```
if ($number<$num)
{
echo “لقد اخترت رقم أصغر من الذي أفكر فيه ; ”
Echo “الرقم الذي كان في مخيلتي هو ;”$num
Echo “<br>”. “يؤسفنا فعلاً أنك لم تنجح، نتمنى أن نقول لك في المرات القادمة ;“
}
```

فإذا لم يتطبق الشرطين فإنه يتركهما ويكتب الكلمة (لقد نجحت) بدون أي كلمات آخرى مثلما كنا نكتب الكلمة (يؤسفنا فعلاً أنك لم تنجح، نتمنى أن نقول لك في المرات القادمة) قبل كلمة (لقد نجحت)، أتمنى أنك قد فهمت جيداً ما أقول.... وتظهر هذه العبارة جلية في الأسطر التالية:

```
?>
لقد نجحت
</body>
</html>
```

على هذا نكون قد صنعنا لعبة كاملة تقوم بإخبار المستخدم عند نجاحه أو خسارته.

معاملات المساواة: == و ===

لقد قمنا باستخدام علامة المساواة الفردية سابقاً في تخزين قيمة في متغير وها نحن نأخذ نوعاً من علامات المساواة وهي علامة المساواة المزدوجة (==) وعلامة المساواة المضاعفة (===).

لقد كنا نستخدم علامة المساواة الفردية أو العادية في تخزين القيم في المتغيرات.

مثال:

```
<?
$m=12;
?>
```

ولكن العلامات التي نتكلم عنها الآن تستخدم في تحديد إذا ما كانت قيمة معينة تساوي قيمة أخرى.

مثال:

```
<?
$m="11";
$u=11;
If ($m==$u)
{
Echo "القيم متساوية";
}
?>
```

لاحظ أن $m متغير حرفي وأن $u متغير رقمي.

إذا كنا نريد إرجاع قيمة إلى متغير نستخدم علامة المساواة العادية (=) وإذا أردنا اختبار متغيرين أو قيمة معينة من أنها متساوية نقوم باختبار القيم بواسطة علامة المساواة المزدوجه (==).

في الـphp4.01 تم إصدار علامة مساواة جديدة تقوم باختبار القيم ولا تعطي القيمة (true) إلا إذا كانت أنواع القيم متساوية وأنواع البيانات في المتغيرات أيضاً متساوية.

مثال (١):

```
<?
$m="11";
$u=11;
If ($m==$u)
{
Echo "القيم متساوية ";
}
?>
```

مثال (٢):

```
<?
$m="11";
$u=11;
If ($m===$u)
{
Echo "القيم متساوية ";
}
?>
```

التوضيح

لاحظ أننا في المثال الأول استخدمنا علامة المساواة المزدوجة لاختبار القيم وكانت القيم متساوية في المتغيرين فتمت طباعة أن القيم متساوية (مع أن نوع البيانات مختلف) ولكن في المثال الثاني عندما استخدمنا علامة المساواة المضاعفة لم تتم طباعة أي شي وذلك لأن القيم متساوية ولكن نوع البيانات مختلف فالمتغير $m حرفي بينما المتغير $u رقمي.

المعاملات: =! و <>

إن عكس علامة المساواة هي علامة عدم المساواة (=!)

مثال:

```
<?
If (5!=99) echo "القيم غير متساوية ";
?>
```

لاحظ أن ٥ لا تساوي ٩٩ لذلك فإن الشرط صحيح (true) لذلك قام بطباعة أن القيم غير متساوية.

إن الضد من علامة (أكبر من) و(أصغر من) هو علامة الـ(<>) وهو يقوم بإرجاع قيمة (true) إذا كانت القيمتين مختلفتين عن بعضهما أي أنه مثل علامة =! تقريباً.

مثال:

```
<?
If (5<>99) echo "القيم غير متساوية ";
?>
```

تطبيق عملي على علامات المساواة وعدم المساواة

قم بفتح محرر النصوص لديك واكتب الكود التالي:

```
<html>
<head></head>
<body>
<Form method =get ACTION= “quiz.php”>
ماهو اسم الرجل الذي يسمي بالفاروق ؟
<br><br>
<input type =”radio” name = “man” value=”عمر”>
عمر بن الخطاب
<br>
<input type =”radio” name = “man” value=”أبو بكر”>
أبو بكر الصديق
<br>
<input type =”radio” name = “man” value=”عثمان”>
عثمان بن عفان
<br>
<input type = submit>
</form>
</body>
</html>
```

احفظها باسم quiz.html...

قم بفتح محرر النصوص لديك واكتب الكود التالي:

```
<html dir=”rtl”>
<head></head>
<body>
<?
```

```
If ($q==" عمر") echo "الإجابة صحيحة";
If ($q!=" عمر") echo "الإجابة خاطئة";
?>
```

قم بحفظه باسم quiz.php وضعهما في مجلد السيرفر.

قم بتشغيل الملف quiz.html

المعاملات المنطقية [AND،OR،NOT]

إن هذه المعاملات المنطقية تتيح لك تنفيذ الكود بعد التحقق من مجموعة شروط
وأيضاً تنفيذ الكود إذا تحقق أكثر من شرط: (AND)
أو تحقق شيء معين من بين عده أشياء: (OR)
ويمكنك مثلاً التحقق من عدم صحة شيء لكي تقوم بتنفيذ شيء آخر: (NOT)

فيمكنك مثلاً أن تقول: إذا كان الجو ممطراً والعاصفة شديدة فلن أخرج من البيت.
ويمكنك أن تقول: إذا كان الجو هادئاً أو لا يوجد أمطار فسأقوم بالخروج إلى المنتزه.
ويمكنك أيضاً أن تقول: إذا لم يكن الجو ممطراً سأقوم بالخروج إلى نزهة.

ولكن عند استخدامك لهذه الدوال عليك مراعاة أن تقوم بجعل هذه الشروط بين قوسين.

المعامل (AND) ونظيره(&&)

يمكننا استعمال المعامل (AND) والمعامل (&&) للتحقق من صحة عدة شروط لتنفيذ شيء معين

مثال(١)

```
<?
$w=10;
$g=12;
IF ($w=10 and $g=12) echo ("لقد تحققت جميع الشروط");
?>
```

مثال (٢)

```
<?
$w=10;
$g=12;
IF ($w=10 && $g=15) echo ("لقد تحققت جميع الشروط");
?>
```

في المثالين السابقين قمنا بعملية التحقق من أكثر من شرط باستخدام المعاملين (&& و and) فعندما تحققت جميع الشروط تم تنفيذ الأمر.... وعندما لم تكن جميع الشروط صحيحة تم تجاهل الأمر.

لاحظ أننا قمنا بجعل الشروط بين قوسين () لكي يعمل الكود بشكل صحيح:

```
($w=10 && $g=15)
($w=10 and $g=12)
```

المعامل (OR) ونظيره (||)

المعامل OR يقوم بالتحقق من عدة شروط وإذا تحقق أي واحد منها فإنه يقوم بتنفيذ الكود ونظيره (||) الذي يقوم بنفس العملية.

مثال (١)

```
<?
$E=100;
$T=8;
IF ($E=14 OR $E=55 OR $E = 10 OR $T=8) echo ("لقد تحقق أحد هذه الشروط");
?>
```

مثال (٢)

```
<?
$E=100;
$T=458;
IF ($E=14 || $E=55 || $E = 10 || $T=8) echo ("لقد تحقق أحد هذه الشروط");
?>
```

إذن عندما تحقق واحد من هذه الشروط تمت طباعة السطر (لقد تحقق أحد هذه الشروط).

ملحوظة: قد لاتكون بتلك الأهمية لكن يجب أن تعرف أن الرموز && و || لها الأسبقية والأفضلية على استخدام AND و OR.

المعامل NOT ونظيره (!)

في الواقع لايمكنك استخدام NOT أبدا لأنها ليست موجودة أصلاً في لغة PHP لكن يمكنك استخدام المعامل (!) كبديل لها فهو يؤدي نفس وظيفتها وهي القيام بالتأكد من أن هناك قيمة غير صحيحة (FALSE) لكي يتم تنفيذ شي معين.

```
<?
$F="رسلان";
IF !($F=="فهمي") echo ("أهلاً بك");
?>
```

في المثال السابق يقوم الـPHP بالتأكد من أن المتغير F$ لايحتوي على القيمة الحرفية (نعمان) ويتم ذلك باستخدام المعامل (!) وعندما يتم التأكد من ذلك يقوم بطباعة السطر (أهلاً بك)

ونشير إلى أننا عندما نقوم باختبار متغير بواسطة المعامل (!) فإن الـPHP إذا وجد المتغير فارغاً أو لم يتم إنشاؤه يعطيه القيمة صفر وهي FALSE.

مثال

```
IF (!($R)) echo (10);
```

استخدام المعاملات <= و =<

من المعلامات المعروفة والمشهورة في الرياضيات هي علامتي (أصغر من أو يساوي) <= أو (أكبر من أو يساوي) >= وهي تستخدم بنفس وظيفتها بالـphp وهي معرفة إذا ما كانت قيمة أصغر أو أكبر من أو تساوي قيمة آخرى، وهذه الأمثلة تعطيك مدخلاً أشمل لفهم هذه الدوال:

```
<?
$t = 15;
If ($t >= 10) echo (“ممتاز”. “<br>”);
$t = 5;
If ($t <= 9) echo (“جيد جداً”);
?>
```

تجميع المعاملات

يمكننا في الشرط أن نتحقق من مجموعة من القيم باستخدام مجموعة من المعاملات، ونقوم بتجميع هذه المجموعات داخل أقواس () مثلما كنا نستخدم سابقاً أكثر من معامل (+، -، /، ❖) باستخدام الأقواس.

وسيبدو ذلك واضحاً وجلياً في مثالنا التالي:

```
<?
$a=10;
$y=5;
$t =29;
If (($a == 10) or ($a==54) and ($y !=25) and ($t >= 11)) echo "تحققت جميع الشروط";
?>
```

ستتم طباعة ١٨ لأن قيمة تجميع التعبير السابق تكون صحيحة ولو قمنا بشرح المثال فسنقوم برؤية القسم الأول وهو:

($a == 10) or ($a==54)

وطبعاً المتغير يحمل القيمة ١٠ فسيكون هذا الجزء صحيحاً.

ثم نقوم برؤية الجزء:

($y !=25) and ($t >= 11)

وطبعاً تم التحقق من جميع الشروط وتمت طباعة الكلمة (تحققت جميع الشروط).

تعدد الشروط [else if و else]

يمكننا استخدام أكثر من هيكلية للعباره if فهناك مثلاً الهيكلية التالية:

```
If condtion is true
{
Excute code
}
Else
{
Excute other code
}
```

وهي تقوم بالتحقق من الشرط فإذا وجدته صحيحاً قامت بتنفيذ الكود الأول وإذا لم تجده صحيحاً ستقوم بتنفيذ الكود الآخر.

مثال

```
<?
$age=10;
If ($age>18)
 {
echo"مرحباً بك في أكبر موقع تجاري إلكتروني";
}
else
{
echo "ممنوع دخول الأطفال الموقع لأنهم لا يملكون المال";
}
?>
```

ويمكننا أيضاً استخدام الهيكلية التالية:

```
If condtion is true
{
Excute code
}
Elseif
{
```

```
Excute other code
}
Else
{
Excute other code
}
```

وهـي تقـوم بتطبيـق أكـثـر مـن شـرط فـإذا لم يكـن أي شـرط مـن الـشروط صـحيحاً سيتم تنفيذ الكود الذي يقع بعد كلمه else.

مثال:

```
<?
$age=10;
 If ($age<=18)
 {
;"مرحباً بك في أكبر موقع تجاري إلكتروني"echo
}
;)elseif)$y >= 44
{
"; لا يوجد مشكلة أيضاً إذا كنت كبيراً echo “
}
else
{
";ممنوع البقيةecho “
}
?>
```

تعشيش العبارات الشرطية

يمكنك تعشيش العبارات الشرطية ، ونعني بتعشيش العبارات الشرطية هـي أن تقـوم بعملية تعشيش الشروط فمثلاً إذا كان شرط ما صحيحاً فإنه يجب أن يكون شرط آخر صحيح لكي يتم حصول شيء معين وغير ذلك.

مثال:

```
<?
$h=”ahmed”;
$f=45;
If  ($h = = “ahmed”)
If ($f= = 45)               {
{
”;الاسم والرقم صحيحانecho “
}
 else
{
 “);الرقم غير صحيحecho (“
}       }
{   else
 “ ; اسم تسجيل الدخول غير صحيحecho “
}
?>
```

هذا مجرد مثال بسيط جداً لتعشيش الدوال الشرطية حيث يقوم بإجراء اختبار على قيمة معينة ثم يقوم عند تجاوز ذلك الاختبار بنجاح بإجراء اختبار ثاني فإذا تم تجاوز الاختبار الثاني تتم طباعة الاسم والرقم صحيحان وإذا لم يتم الاجتياز تتم طباعة عبارة الفشل في الاجتياز.

تطبيق عملي

سنقوم في هذا التطبيق بصناعة مسابقة بسيطة نستخدم فيها ما تكلمنا عنه سابقاً.

١- قم بإنشاء ملف Msabqa.html.

٢- قم بكتابه الكود التالي فيه:

```
<html>
<body>

<form method="POST" action="msabqa.php" dir="rtl">
 من هو أول الخلفاء الراشدين <br>
```

```
<p><br><input type="radio" value="abubaker"
<br><input type="radio" value="3mar" أبو بكر الصديقname="s">
عمرname="s">
<br><input type="radio" value="3thman" checked
<br><br><br>عثمانname="s">

</p>

">    <input إرسال  <p><input type="submit" value="
"></p>حـــــذفtype="reset" value="
</form>

</body><html>
```

قم بفتح ملف وقم بتسميته msabqa.php

```
<?
<html dir = “rtl”>
If $s == “3mar”{
الإجابة صحيحة
}
else
{
”; الإجابة خاطئة echo “
}
?>
```

العبارة Switch

```
Switch (VARIABLE) {
:CASE THING1
Excute code ;
            break;
:CASE THING2
Excute code ;
break;
 Default;
Excute code ;

  }
```

تقوم العبارة بنفس عملية العبارة if ولكن بهيكلية أسهل ومحببة أكثر وتتيح لك اختبار قيمة متغيرة وإجراء أكثر من اختبار عليه.

break;

تقوم بالخروج من عبارة معينة مثل switch و if والذهاب إلى الأوامر والعبارات التي بعدها.

EXIT;

تقوم بعملية الخروج من الكود نهائياً ولا تطبق أي أوامر بعدها ، وفي الأمثلة التوضيحية التالية ستجد أن break; تخرج من العبارة فقط (Statement) بينما الـexit; تقوم بالخروج من كامل الكود (code).

مثال:

```
<?
$s=10;
if ($s=10) {
echo “number=10”;
```

```
exit;
}
elseif ($s<11) {
  echo "number is less than 11"
{
echo "hello";
?>
```

مثال:

```
<?
$s=10;
if ($s=10) {
echo "number=10";
break;
}
elseif ($s<11) {
  echo "number is less than 11"
{
echo "Hello";
?>
```

Defualt;

إذا لم تصلح جميع الحالات (Cases) في العبارة (Switch) فسيتم تنفيذ الأوامر التي تقع بعد هذه الكلمة وهي تؤدي نفس عمل else تقريباً في العبارة if.

مثال (١)

```
<?
$g= "ahmed";
Switch ($g) {
Case "ahmed":
 Echo "مسموح ";
Break ;
  Case "khaled ":
```

```
Echo “ممنوع; “
Break ;
Case “salem“:
Echo “ممنوع; “
Break ;
Case “Mohmed “:
Echo “مسموح; “
Break ;
Default ;
Echo “لقد أدخلت اسماً غير صالح;”
}
?>
```

مثال (٢)

```
Switch ($g) {
Case $g>50:
Echo “كبير;“
Break ;
Case  40 :
Echo “لا بأس; “
Break ;
Case ($g<15):
Echo “أطفال ممنوع; “
Break ;
Case 30:
Echo “مسموح; “
Break ;
}
```

لاحظ أننا عند اختبارنا لنصوص نحتاج إلى علامتي تنصيص مزدوجة وعند الأرقام فإننا لا نحتاج إلى ذلك.

تطبيق عملي

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم age.html

```
<html>
<form method=post action=”age.php”>
كم عمرك ؟
<br>
<input type=”text” name = “g”>
<input type=submit value=”إرسال”>
</form>
</html>
```

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم age.php

```
<?
Switch ($g) {
Case $g>50:
Echo “كبير“;
Break ;
Case 40 :
Echo “لا بأس“ ;
Break ;
Case ($g<15):
Echo “أطفال ممنوع“ ;
Break ;
Case 30:
Echo “مسموح“ ;
Break ;
}
?>
```

الشرح

تقوم العبارة Switch باختبار قيمة متغير ما ويمكنك إجراء أكثر من افتراض عليه ويجب عليك كتابة الكلمة ;break لكي تقوم بإيقاف تنفيذ العبارة switch فمثلاً لو قمت بكتابة الكود التالي:

```
<?
$g=40
Switch ($g) {
Case $g<50:
Echo "1 ";

: Case 40
Echo "2" ;
}
?>
```

فإذا أدخل المستخدم الرقم ٤٠ فسيتم طباعة الرقمين واحد واثنين كلاهما وذلك لأنك لم تقم بإيقاف العبارة فاكملت التحقق وطبقت جميع العمليات المطلوبة.

التخلص من وسوم الـ html

إذا قمت بوضع مربع نص وأردت من المستخدم كتابة شي فيه فإنه يستطيع إدخال أي شيء ولنفترض أنه كتب في مربع النص كالتالي:

```
I am <b>ahmed</b> …
```

فسيقوم المتصفح بعرضها بعد معالجتها كالتالي:

```
I am ahmed ….
```

ولنقم بتطبيق عملي على ذلك

قم بفتح محرر النصوص واكتب الكود التالي واحفظه باسم htmlch.html

```
<html dir="rtl">
<form method=post action="html.php">
أدخل اسمك الكريم
```

```
<br>
<input type="text" name = "fname">
<input type=submit value="إرسال">
</form>
</html>
```

قم بفتح محرر النصوص واكتب الكود التالي واحفظه باسم html.php

```
<?
Echo "هذا هو الشكل الطبيعي للعبارة عند طباعتها";
Echo "<br>". $fname;
?>
```

قم بوضع الملفات في مجلد السيرفر ثم قم بتشغيل الملف htmlch.html واكتب في مربع النص أي شيء وضعه بين وسوم html

مثال:

I am <i>alfareees</i>

ستجد أنه قد تم التعامل مع الوسوم كـ html وليس كنص عادي ولكي تعرضها كنص عادي فإنك تقوم باستخدام الدالة

HtmlSpecialChars();

حيث أنها ستقوم بمعاملة كود الـhtml كنص عادي وطبيعي تماماً.

إذا نقوم بتعديل ملف الـhtml.php ليصبح كالتالي:

```
<?
$fname = HtmlSpecialChars($fname);
Echo "هذا هو الشكل بعد استخدام الدالة";
Echo "<br>". $fname;
?>
```

الفصل الرابع
التكرارت والمصفوفات

لقد أخذنا في البحث السابق شيئاً من أساسيات البرمجة وهو الدوال الشرطية وصناعة القرارات والآن نحن نتجه إلى شيء يحب جهاز الكمبيوتر عمله وهو التكرارت والمصفوفات.

في الواقع قد يكون لديك يومياً شي تفعله بشكل مستمر مثل الإفطار في الصباح الباكر والنوم مساء، انك تستمر على هذا الروتين دائماً.... نحن نسمي هذا الشي في لغة البرمجه التكرار.

هناك شي آخر يسمي المصفوفات... في الواقع قد يحتوي درج مكتبك الخاصة بالكتب على عدة أدراج الدرج الاول منها يحتوي على الكتب الإسلامية والدرج الثاني منها يحتوي على الكتب الرياضية والدرج الثالث يحتوي على كتب الرياضيات... أو لنفرض أنك مدرس في إحدي المدارس ولديك جدول للحصص ففي الحصة الأولى لديك مثلاً تدريس مادة الرياضيات.... والحصة الثانية لديك تدريس مادة العلوم والثالثة لديك تدريس مادة الكيمياء.... إن حصصك مرتبة بشكل معين مع أنها كلها تسمي حصص إلا أن كل حصة تختلف عن الأخري في المادة ! وهي مرتبه بشكل تصاعدي (الحصة الاولى، الثانية، الثالثة.....).

نسمي هذه التقنية بالمصفوفات.... المصفوفات عبارة عن متغير اسمه ثابت ولها اكثر من قيمة وكل قيمة لها رقم معين ولكي تحصل على القيمة فانك تكتب المتغير ثم رقم القيمة التي فيه، لايشترط أن تكون هذه القيم متسلسلة فقد يكون هناك قيمتين ولكل قيمة رقم يختلف تماماً ويبعد كل البعد عن القيمه الثانية مثال رقم ١ و ٢٥٨ كلاهما مختلف تماماً ويبتعد كل البعد عن الاخر.

إن دمج ميزة التكرارات مع المصفوفات يساعدك على توفير عدد الأسطر للكود ويساعدك على صنع أشياء عجيبة في أقل عدد ممكن من الأسطر.

التكرارات

التكرارات عبارة عن تكرار أمر معين بعدد معين من المرات ولقد اخذنا سابقاً الدوال الشرطية أو العبارات الشرطية بالأصح فوجدنا أن الكود الذي نكتبه في العبارات الشرطية لا تتنفذ إلا عندما يكون الشرط صحيحاً
أيضا التكرارت فهي تختبر الشرط فإذا كانت قيمته صحيحة فإنها تقوم بعمل الكود المطلوب ثم تقوم بإعادة اختبار القيمة فإذا كان صحيحاً فإنها تقوم بإعادة تنفيذ الكود وهكذا ، أما عندما لا يكون الشرط صحيحاً فإنها تتوقف عن تنفيذ الكود ويتم اكمال البرنامج بشكل عادي... هناك ثلاثة أنواع من التكرارات.
إن أول دالة نقوم بأخذها في البداية هي الدالة while

التكرار while

لقد قمنا بأخذ التكرار while لأنه بسيط جداً وصيغه هذا التكرار هي:

```
While (condition شرط )
{
code
}
```

مثال:

```
<?
$d =10 ;
while ($d<15)
{
echo “$d <br>”;
$d++;
}
?>
```

سيقوم الـPHP أولاً بإعطاء المتغير $d القيمة ١٠ ثم يقوم بعد ببدء التكرار while فإذا كان الشرط صحيحاً (وهو أن المتغير أصغر من الرقم ١٥) فإنه يقوم بتنفيذ الكود الذي بين الأقواس وعمل هذا الكود أن يقوم بطباعة المتغير ثم يقوم بإضافة واحد على القيمة الموجودة في المتغير $d ثم بعد ذلك سيتم اختبار الشرط مرة ثانية فإذا كان صحيحاً فستتم نفس العملية حتي يكون الشرط غير صحيح فيتوقف عندها التكرار ويتم إكمال الكود الذي يقع بعد الأقواس.

إذا لم تقم بوضع حد للتكرار فلن يتوقف التكرار وقد يكون لانهائي....

مثال:

```
<?
$d =10 ;
while ($d<15)
{
echo “$d <br>”;
}
?>
```

ستتم طباعة الرقم ١٠ ولن يتوقف التكرار لأن الشرط صحيح دائماً وليس هناك ما يوقفه بينما في الكود السابق استطعنا إيقاف الكود بسبب أننا كنا نضيف واحد على القيمة الموجودة في المتغير وكلما تمت إعادة اختبار الكود كلما تغيرت القيمة حتى يصبح الشرط غير صحيح بسبب أن $d أكبر من ١٥.

التكرار do - while

هذا التكرار يعمل بنفس طريقة التكرار الأول إلا أنه يوجد بعض الاختلافات البسيطة وصيغته كالتالي:

```
do
code
while (condition شرط);
```

مثال:

```
<?
$f=15 ;
do
{
echo "$f";
$f ++
}
while () ;
```

سيقوم التكرار بتنفيذ السطر الموجود بين القوسين أولاً ثم يقوم بتنفيذ باختبار الشرط فإذا كان الشرط صحيحاً قام بإعادة العملية الموجودة بين القوسين وهي إضافة واحد على المتغير **$f** وهكذا حتي يكون الشرط غير صحيح فيتم التوقف.. لاحظ أننا في التكرار الأول قمنا باختبار الشرط قبل صناعة أي عمل بينما في التكرار الثاني قمنا بتنفيذ الكود أولاً ثم قمنا بإجراء الاختبار.

التكرار FOR

يختلف هذا التكرار عن سابقيه لكن وظيفته هي نفس وظيفتهما وهي تكرار الأوامر عند حصول شيء معين

الصيغة:

```
For (counter عداد ; test value اختبار القيمة ; set counter أداء عمليه على العداد)
{
code شيفرة
}
```

مثال:

```
<?
For ($u = 18 ; $u>10 ; $u--)
{
echo $u;
}
?>
```

يتكون هذا التكرار من ثلاثة أقسام.... القسم الأول نضع فيه متغير يحتوي على قيمة حيث سيبدأ التكرار العمل من عند هذه القيمة والقسم الثاني نكتب فيه الشرط الذي سيقوم التكرار بفحصه (والذي هو كالمعتاد اختبار لقيمة المتغير في القسم الأول) والقسم الثالث نضع فيه العمل الذي سيجري على المتغير عند كل تكرار ثم نقوم بكتابة الكود الذي سيقوم بتنفيذ التكرار بين القوسين.

كأننا نقول للـphp بشكل عام أن يقوم في البداية بإعطاء المتغير $u القيمة ١٨ وقبل أن يقوم بتنفيذ الكود عليه أن يقوم بتحليل الشرط فإذا كان الشرط صحيحاً فإنه يقوم بإنقاص واحد من المتغير $u ويتم تنفيذ الكود حتى يصبح المتغير$u قيمته ٩ فيقوم الـPHP آنذاك بالخروج من التكرار والذهاب إلى الكود الذي يلي القوسين.

المصفوفات

لقد قمنا بتعريف المصفوفات سابقاً بشكل بسيط وحان الوقت الآن لنعرفها ونعرف كيفية عملها.

المصفوفات عبارة عن متغير وهذا المتغير يحتوي على أكثر من قيمة أو عنصر (element) وكل عنصر له فهرسة (Index) تبدأ هذه الفهرسة من الصفر إذا لم تقم بتحديدها.

مثال:

```
<?
$A[ ] = “alfareees”;
$A[ ] = 13;
?>
```

في هذا المثال سيقوم الـPHP بإعطاء الفهرسة تلقائياً فسيقوم بوضع الرقم فتصبح المتغير فهرسته كالتالي:

```
$A[0] = “alfareees”;
$A[1] = 13;
```

إننا لم نقم بإدخال هذه الأرقام من تلقاء أنفسنا ولكن الـPHP قام بوضعها مع أنه يمكننا أن ندخلها بشكل عادي فمثلاً لو كتبنا:

```
<?
$A[0]= “alfareees”;
$A[1] = 13;
?>
```

سيقوم الـPHP بأخذ الفهرسة المعتمدة ولن يضع أي فهرسة أخرى يمكننا أيضاً أن نكتب أي فهرسة ولا نعتمد على الترتيب في الأرقام.

مثال:

```
<?
$A[10 ] = “alfareees”;
$A[ 25] = 13;
?>
```

هل لاحظت أيضاً أننا لم نقم بتعريف نوع متغيرات المصفوفة وقام الـPHP بتعريفها تلقائياً بدلاً منا فمرة استخدمنا قيمة حرفية ومرة استخدمنا رقماً ورغم ذلك فلم يقم الـPHP بعمل أي اعتراض إضافة إلى ذلك فإن الـPHP يقوم بتحديد عدد عناصر المصفوفة تلقائياً فهو يعرف مثلاً من المثال السابق أن عدد عناصر المصفوفة الكلي هو عنصرين.

يمنحنا الـPHP ميزة أخري وهي عدم التقيد بالأرقام في الفهرسة فمثلاً يمكننا استخدام حروف عادية.

مثال:

```
<?
$A[“a” ] = “alfareees”;
$A[“b” ] = 13;
?>
```

لاحظ أننا استخدمنا القيم الحرفية ولم يعترض الPHPJ بتاتاً ويمكننا طباعة أي عنصر من عناصر المصفوفة بكل بساطة.

مثال:

```
<?
$r [“aa”] = “ahmed ali”;
$r [1] = 13273;
$r [20] = 13273;
echo $r[aa];
echo $r[20];
];"aa"echo $r[
?>
```

لا فرق بين أن نكتب النص الحرفي (aa) بين علامتي تنصيص عند الطباعه وعند كتابته بدون علامات تنصيص... سيقوم الـPHP بمعرفة ذلك تلقائياً.

يمكننا تعريف المصفوفات أيضاً بطريقة أخرى

```
$variable = array (elements) ;
```

مثال:

```
<?
 “alfarsi”); ، “salem”، “ali”،$t =array (“ahmed”
echo $t [0];
?>
```

يقوم الـPHP بإعطاء كل عنصر من عناصر المصفوفة رقم فهرسة فتصبح كالتالي:

Index الفهرسة	Element العنصر
٠	Ahmed
١	Ali
٢	Salem
٣	alfarsi

إذن القيمة التي سيطبعها الـPHP في النهاية هـي ahmed، لاحظ أن الـPHP قام بإعطاء رقم الفهرسة وقام بالبدء من الصفر ولكن يمكننا جعل الـPHP يبدأ الفهرسة من الرقم واحد كالتالي:

```
<?
$r = array (1=>”ahmed”، “ali”، ”salem”، “alfarsi”);
?>
```

عند تعريفك لرقم الفهرسة للقيمة الأولى سيقوم الـPHP بإعطاء أرقام فهرسة بشكل تسلسلي، عندئذ ستصبح الفهرسة كالتالي:

Index الفهرسة	Element العنصر
١	ahmed
٢	Ali
٣	salem
٤	alfarsi

هناك طريقة لتكون أيضاً الفهرسة هي عبارة عن حروف:

```
<?
$r = array (“ss”=>”ahmed”، “sf”=> “ali”، “da”=>”salem”، “bv”=> “alfarsi”);
?>
```

عندئذ ستصبح الفهرسة كالتالي:

Index الفهرسة	Element العنصر
Ss	Ahmed
Sf	Ali

Salem	Da
Alfarsi	Bv

عندما نريد تغيير أي عنصر في المصفوفة فيمكننا عمل ذلك ببساطة.

مثال:

```
$r [ss]= “ المياء”;
```

لاحظ أننا قمنا بتغيير القيمة من (ahmed) إلى (المياء)....طريقة بسيطة أليس كذلك:

قراءه المصفوفات واستخراج القيم

تكلمنا سابقا عن التكرار For

يمكننا استخراج عناصر مصفوفة وطباعتها ببساطة وتوفير وقت عن طريق التكرارات

لنفرض أن لديك هذه المصفوفة:

```
<?
$people =array (“ahmed”، “ali”، “salem”، “alfarsi”);
?>
```

وأردت أن تطبع أسماء جميع الأشخاص المتواجدين فيها

أولاً نحن نعرف أن المصفوفة إذا لم نقم بتعريف رقم فهرسة لها فإن الـPHP يقوم ببداية فهرستها من الصفر وعلى ذلك فإن رقم العنصر الأول ٠ ورقم العنصر الرابع ٣... على ذلك يمكننا بكل بساطة كتابة الكود التالي الذي يقوم بطباعة المصفوفة كالتالي:

```
<?
$people =array (“ahmed”، “ali”، “salem”، “alfarsi”);
echo “$people[0]. <br>”;
```

```
echo “$people[1]. <br>”;
echo “$people[2]. <br>”;
echo “$people[3]. <br>”;
?>
```

لنفرض أن لديك ثلاثين أو ثلاثة آلاف اسم في مصفوفة ألن تبدو هذه الطريقة متعبة قليلا !!!

هناك طريقة أخري وهي عن طريق التكرارات.

لنفرض أننا أردنا كتابة تكرار يقوم بطباعة الأرقام من واحد إلى عشرة فإننا نستطيع كتابة التكرار بالشكل التالي:

```
<?
For ($I=1;$I<11;$I++)
{
Echo "$I <br>";
}
?>
```

والآن لنقل أننا نريد طباعة الأربعة عناصر في المصفوفة كل ما علينا هو إجراء عملية بسيطة على الكود لكي يتم ذلك:

```
<?
 "alfarsi"); ، "salem"، "ali"،$people =array ("ahmed"

For ($I=0;$I<4;$I++)
{
Echo "$people[$I] <br>";
}
?>
```

لاحظ أننا بدأنا العداد بالقيمة صفر ثم اشترطنا أن يكون أقل من ٤ لأن آخر عنصر في المصفوفة رقم فهرسته ٣ ثم قمنا بجعله يزداد بقيمة ١ لأننا نريد طباعة جميع

عناصر المصفوفة وقمنا بوضع رقم العداد في خانة الفهرسة وعلى ذلك سيتم في كل تكرار طباعة عنصر المصفوفة الذي فهرسته تساوي رقم العداد.

لقد تكلمنا سابقاً في فصل النماذج عن إخراج القيم من قائمة على شكل مصفوفة.

مثال:

```
<form action = “array.php” method = post>
ما هو مشروبك المفضل ؟
<br>
<select name = “a[]” multiple>
<option>شاي</option>
<option>قهوة</option>
<option>كابتشينو</option>
<option>متة</option>
<option>برتقال</option>
</select>
<br>
<input type=submit value = “ لذيذ ” >
</form>
```

في ملف الـ array.php اكتب:

```
<html>
لقد قمت باختيار التالي:
<?
For ($I=0;$I<4;$I++)
{
Echo "$a[$I] <br>";
}
?>
</html>
```

لقد عرضنا في القائمة خمسة عناصر... لاحظ أننا وضعنا في اسم المتغير للقائمة قوسين [ ] لكي يتعرف الـhtml على أنه سيتم تخزين البيانات تلقائياً بعد ذلك قام الـPHP بفهرسة العناصر التي تم إرسالها من قبل العميل سواء كانت ثلاثة أو أربعة ولكنها بالطبع لن تزيد على خمسة.... على ذلك سيكون آخر رقم تنتهي به المصفوفة هو ٤.

أتوقع أنك الآن بدأت تحب المصفوفات.... يمكننا صناعة القائمة عن طريق المصفوفة أيضاً....

مثال:

```
<form action = "list.php" method = post>
من هو صديقك المفضل ؟
<br>
<select name = "s" >
<?
$shrab =array("شاي"،"قهوة"،"كابتشينو"،"متة"،"برتقال");
For ($k=0;$k<4;$k++)
{
echo "<option>".$shrab[$k]."</option>";
}
?>
</select>
</form>
```

عند اختيار المستخدم للقيمة سيتم وضعها في المتغير $s يمكنك مراجعة درس النماذج لكي تفعل ذلك، هذا المثال يقوم بصناعة مصفوفة للمشروبات ثم يقوم بإخراجها في قائمة مما يوفر علينا الوقت في كتابة الكود فلو كان لديك مثلا حوالى مئة دولة فيمكنك مثلا وضعها في مصفوفة وبعد ذلك بناء القائمة التي سوف تقوم ببناء القائمة التي ستحتوي على هذه الدول عن طريق المصفوفات والتكرارات.

قم بحفظ التغييرات في ملف إمتداده php وقم بكتابة الملف list.php اعتمادا على معلوماتك السابقة في درس النماذج.

الدالة key

لنفرض أن لدينا مصفوفة مكونة من عنصرين:

مثال:

```
$s= array ("على"، "محمد");
```

الآن لنضف إليها هذه السطور

```
<?
$s= array ("على"، "محمد");
$t=key ($s);
echo $t;
?>
```

يقوم الأمر key بإيجاد رقم الفهرسه (index) العنصر النشط حالياً.... وهو الرقم صفر حيث أننا لم نضع فهرسة وهذه هي الفهرسة التي وضعها الـPHP تلقائياً عندما لم نضع فهرسة... قد تحيرك كلمة النشط لكن ستعرف أننا نستطيع التجول بين عناصر المصفوفة لاحقاً.

قد يكون رقم الفهرسة حروف أو كلمات

مثال:

```
<?
$s= array ("ع"=>"علي"، "م"=>"محمد");
$t=key ($s);
echo $t;
?>
```

الدالة ()current

تقوم الدالة current بإيجاد القيمة لعنصر المصفوفة الحالي (index value).

مثال:

```
<?
$s= array (“ع”=>”علي”،”م”=>”محمد”);
$p=current ($s);
echo $p;
?>
```

في المثال السابق قمنا بإيجاد القيمة الحالية للعنصر النشط.... لاحظ أننا أوجدنا بالأمر key رقم الفهرسة بينما أوجدنا بالأمر current القيمة للعنصر المفهرس.

كيف يمكننا تنشيط العناصر الأخرى للمصفوفة ؟!

يمكننا ذلك عن طريق الدالتين   ()next و prev اللتان تقومان بالتجول بين عناصر المصفوفة..... لنفرض أن لدينا مصفوفة تتكون من ثلاثة عناصر

مثال:

```
<?
$s= array (“ع”=>”علي”،”م”=>”محمد”،”ا”=>”أحمد”);
echo key($s)."<br>";
echo current($s)."<br>";
?>
```

لقد قمنا في هذا المثال بطباعة قيمة رقم الفهرسة للعنصر الحالي وقيمته (اقصد برقم الفهرسة الحرف(ع) وأقصد بالقيمة (على)).... لنقم الآن بالتجول بين عناصر المصفوفة ولنر نتيجة الطباعة.

مثال:

```
<?
$s= array (“ع”=>”علي”،”م”=>”محمد”،”ا”=>”أحمد”);
next($s);
echo key($s)."<br>";
echo current($s)."<br>";
```

```
?>

<?
$s= array ("ع"=>"علي"، "م"=>"محمد"، "ا"=>"أحمد");
next($s);
next($s);
echo key($s)."<br>";
echo current($s)."<br>";
?>
```

لاحظ أننا كتبنا الدالة next() قبل أن نقوم بالانتقال لكي يتم تنشيط العنصر الثاني في أول مثال ولتنشيط العنصر الثالث في ثالث مثال (ولاحظ أننا كتبنا next() مرتين).

يمكننا الرجوع لتنشيط العنصر السابق بوضع الدالة prev() فمثلاً يمكننا تعديل المثال التالي:

```
<?
$s= array ("ع"=>"علي"، "م"=>"محمد"، "ا"=>"أحمد");
next($s);
next($s);
prev($s);
echo key($s)."<br>";
echo current($s)."<br>";
?>
```

فسيقوم الـPHP في هذه الحالة بطباعة العنصر الثاني وليس الثالث لأنه تم التراجع خطوة عن طريق prev()

ماذا سيحصل إذا قمنا بإضافة عنصر على مصفوفة غير محدودة الفهرسة ؟!

لنفرض أن لدينا مصفوفة وأضفنا إليها عنصراً غير محدد الفهرسة. مثل:

```
<?
$s= array (12=>"علي"، 5=>"محمد"، 44=>"أحمد");
$s[ ]= "هشام";
Next($s);
```

```
Next($s);
Next($s);
Echo key ($s)."<br>";
Echo current($s)."<br>";
?>
```

سيقوم الـPHP ببساطة بالبحث عن أكبر رقم فهرسة وبعد ذلك يبدأ بإعطاء الفهرسة تسلسلاً بعده فإذا كانت أرقام الفهرسة حروفاً بدأ من الصفر في إعطاء الرقم.. ولاحظ في هذا المثال بأنه قام بإعطاء العنصر الرقم ٤٥ لأن أكبر عنصر في المصفوفة هو ٤٤ وعلى ذلك قام بإعطاء الأرقام تسلسلاً بعد هذا الرقم.

الدالة List و Each

لنفرض أنك قد قمت بصنع مصفوفة غير مفهرسة بالترتيب

مثال:

```
<?
$s= array (12=>"علي"، 5=>"محمد"، 44=>"أحمد");
?>
```

على ذلك دعنا نخبرك بخبر سار وهو أنك تستطيع أن تجعل حياتك مع PHP أسهل من حياتك مع نفسك !

```
While (list(ارقام الفهرسة Index ، قيمة العنصر Element value) =each (array)
```

تستطيع بواسطة هاتين الدالتين وعن طريق التكرار while استخراج جميع العناصر الموجودة في المصفوفة

```
While (list($e ، $r) = each ($s))
{
echo "<br> $e<br> $r";
}
```

أولاً أنت تقوم بتسمية متغيرين واحد منهما لرقم الفهرسة ($e) والثاني للعنصر ($r) ويمكننا تسميتهما بأي اسم وفي حالة ما إذا أردنا عرض العنصر فقط أو معرفة العنصر فقط فيمكننا حذف ($e) ولكننا لا نحذف الفاصلة

```
$r) = each ($s))،While (list(
{
echo “<br> $e<br> $r”;
}
```

لنعـد إلى المثـال الـذي فيـه رقـم الفهرسـة والعنـصر... سـيقوم التكـرار بوضـع رقـم الفهرسـة (الـذي قـد يكـون نـصياً) في المتغير $e وسيـضع قيمـة العنـصر الـذي رقـم الفهرسـة لـه هـو $e في المتغير $r ثـم سـيقوم بطباعـة العناصـر حتـى ينتهـي منهـا جميعها...

ملاحظـة مهمـة: إذا لم تقـم بتعريـف فهرسـة للمـصفوفة (حـروف أو أرقـام أيـا كـان) فسيتم استخدام العناصر عندما يطلب التكرار الفهارس.

مثال:

```
<?
"tewr");،"terhfgfd"،$e=array("fsda"
$V)=each($e))،While (list ($I
{
echo "<br>$e[$I]";
}
?>
```

لاحـــظ مـــع أننـــا طلبنـــا طباعـــة الفهرســـة (index) إلا أنـــه تم أخـــذ العناصـــر (elements) بدلاً من الفهرسة.

يمـكننـا بواسـطة هـذه الدالـة صناعـة أشـياء مفيـدة وكـمثـال لـذلك لنفـرض أن لـدينا مصفوفة أرقام هواتف ونريد أن نخرج هذه المصفوفة علـى جـدول html فسنستطيع صناعة هذا الجدول عن طريق التكـرار السابق بكل سهولة.

مثال:

```
<table align='center' dir = "rtl" border="1" width="100%"
cellspacing="0" bordercolorlight="#000000" bordercolordark="#000000"
bordercolor="#000000">
<tr>
</td>الاسم     <td align='center'>
```

```
</td>رقم التلفون     <td align='center'>
</tr>
<?
$s = array (658=>"عادل"،٤٥٦٥٤٦=>"سالم"، ٤٦٥٨٧٣);
While (list($e، $r) = each ($s))
{
echo "<tr><td align='center'>". $r. "</td><td align='center'>".$e.
"</td></tr>";
}
?>
</table>
```

أرايت كيف استخرجنا جميع أرقام التلفونات في جدول بواسطة تكرار بسيط، يمكنك صناعة الأكثر واختصار الكثير من الوقت، على ذلك إذا كانت المصفوفة تحتوي على المئات من الأرقام بواسطة هذا الكود بدلاً من أن تكتب الكود على شكل html وتكتب البيانات وتتعب نفسك.

يمكنك أيضاً معرفة عدد العناصر في مصفوفة معينة إذا كنت تريد معرفة عددها وذلك بالطريقة التالية:

```
<?
$s= array (12=>"علي"، 5=>"محمد"، 44=>"أحمد");
$S=0;
While (list($E، $r) = each ($s))
{
$S++;
}
ECHO "عدد عناصر المصفوفة ". $S++;
?>
```

فرز المصفوفات

هناك العديد من الدوال التي يوفرها لنا الـPHP لفرز المصفوفات. نحن سنأخذ نظرة عن الخمسة دوال الأكثر استخداماً:

الدالة ()Sort

هذه الدالة من أساسيات فرز المصفوفات وهي جداً أساسية وهي تقوم بأخذ محتويات المصفوفة ومن ثم تقوم بفرزها هجائياً اعتماداً على الأحرف الكبيرة أولا ثم الصغيرة.. تتطلب هذه الدالة اسم المصفوفة التي سيتم عليها الفرز

```
Sort (ArrayName);
```

إذا قمنا بإنشاء مصفوفة بالشكل التالي:

```
$NaNo=array (“ali”،“salem”،“hythem”،“Khaled”،“Ammar”،“Hesham”);
```

فإذا أردنا فرزها عن طريق الدالة ()sort فإننا نقوم باستخدامها كالتالي:

```
<?
$NaNo=array (“ali”،“salem”،“hythem”،“Khaled”،“Ammar”،“Hesham”);
sort($NaNo);
While (list($e، $r) = each ($NaNo))
{
echo “<br> $e<br> $r”;
}
?>
```

لاحظ أنه عند تنفيذك للمثال ستجد أن الـPHP قام بالفرز اعتماداً على الأحرف الكبيرة أولاً ثم قام بالفرز بعدها اعتماداً على الأحرف الصغيرة.

الدالة ()Arsort

هذه الدالة تعمل نفس عملية الدالة ()sort ولكن هناك اختلاف بسيط فمثلاً لو كتبنا المصفوفة كالتالي:

```
“kh”=> “khaled”);  ،$NaNo=array (“ad”=>”ahmed”
```

وأردنا فرزها وطباعة الفهارس والقيم كما في المثال التالي:

```
<?
 “kh”=> “khaled”);  ،$NaNo=array (“ad”=>”ahmed”
sort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

قارن ناتج المثال السابق مع هذا المثال:

```
<?
 “kh”=> “khaled”);  ،$NaNo=array (“ad”=>”ahmed”
asort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

اعتقد انك قد عرفت الفرق ففي المثال الأول قامت الدالة **sort** باستبدال الحروف بأرقام في الفهرسة أما في المثال الثاني فقد تم وضع الحروف كما هي وتم فرزها كما تفعل الدالة **sort** في الفرز.

باختصار لايوجد فرق بين **sort** و **asort** إلا في أن الدالة **sort** تستبدل فهرسة الحروف بأرقام.

الدالة ()Rsort و arsort

تقوم بنفس عمل **sort** و **asort** ولكن بشكل عكسي جرب الأمثلة التالية:

مثال:

```
<?
 “kh”=> “khaled”);  ،$NaNo=array (“ad”=>”ahmed”
rsort($NaNo);
```

```
$r) = each ($NaNo))،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

مثال:

```
<?
 “kh”=> “khaled”); ،$NaNo=array (“ad”=>”ahmed”
arsort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo “<br> $e<br> $r”;
}
?>
```

ستجد أن الدالة **rsort** تقوم بنفس عملية الدالة **sort** ولكن بشكل عكسي.

أيضاً الدالة **arsort** تقوم بنفس عملية **asort** ولكن بشكل عكسي.

يمكنك استعمال كل هذه الدوال في الفرز مع الحروف العربية (إذا كان السيرفر يدعم اللغة العربية)

قم بتطبيق المثال التالي:

```
RSORT()
<?
"); أحمد "kh"=> "،"سالم$NaNo=array ("ad"=>"
rsort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ARSORT()
<?
"); أحمد "kh"=> "،"أحمد$NaNo=array ("ad"=>"
arsort($NaNo);
$r) = each ($NaNo))،While (list($e
```

```
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ASORT()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
asort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
SORT()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
sort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
```

الدالة ksort

تكلمنا سابقاً عن طريقة فرز المصفوفات ولكن نريد أن نلفت نظرك أننا كنا نعتمد على العنصر في الفرز (element) ولكن هذه الدالة تقوم بالاعتماد على رقم الفهرسة في الفرز (index)

مثال:

```
<br>--------<br>
asort()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
asort($NaNo);
$r) = each ($NaNo))،While (list($e
{
```

```
echo "<br> $e<br> $r";
}
?>
<br>--------<br>
ksort()
<?
"); جمال "kh"=> "،"هاشم$NaNo=array ("ad"=>"
ksort($NaNo);
$r) = each ($NaNo))،While (list($e
{
echo "<br> $e<br> $r";
}
?>
```

لقد اعتمد الـphp على index ولم يعتمد على الـelement في الفرز.

دوال المصفوفات الإضافية

هناك الكثير من الدوال التي يمنحنا إياها الـPHP للتعامل مع المصفوفات والتي لا يكفي الوقت لذكرها الآن وسنقوم بشرح أهم دالتين والمستخدمتين بكثرة وهي **array_push()** و **array_pop()**

لنفرض أننا قمنا بإنشاء مصفوفة بالشكل التالي:

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
?>
```

وأردنا أن نضيف عنصراً جديد لها فقمنا بالتالي:

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
$saher[ ]=”Alfarees”;
?>
```

انظر إلى العنصر الأخير الذي سيعطيه الـPHP رقم الفهرسة (index) وسيكون رقم فهرسته هو 86.

نريد أن نلفت نظرك بأننا نستطيع عمل إضافة لعنصر على المصفوفة بطريقة أخرى وهي عن طريق الدالة array_push() كالتالي:

```
array_push (ArrayNameاسم المصفوفه، Elemnt1، Elemnt2، Elemnt3، .....)
```

نضع في القسم الأول من الدالة اسم المصفوفة التي نريد إضافة العنصر لها ونضع في القسم الثاني عنصر واحد أو أكثر وهي التي سيتم إضافتها للمصفوفة.

مثال:

```
<?
$saher[5]="salem";
$saher[85]="khaled";
$saher[35]="mohmed";
$saher[19]="hajeer";
array_push ($saher، Alfarees)
?>
```

مثال:

```
<?
$saher[5]="salem";
$saher[85]="khaled";
$saher[35]="mohmed";
$saher[ 19]="hajeer";
array_push ($saher، Alfarees، salem، sameer، thamer)
?>
```

ولو أردنا حذف مثلاً عنصر من المصفوفة فإننا نقوم بتعريف المصفوفة من جديد أو يمكننا استخدام الدالة array_pop التي تقوم بحذف آخر عنصر من المصفوفة والتي تتطلب فقط اسم المصفوفة

```
Array_pop(ArrayNameاسم المصفوفة)
```

مثال:

```
<?
$saher[ 5]=”salem”;
$saher[ 85]=”khaled”;
$saher[ 35]=”mohmed”;
$saher[ 19]=”hajeer”;
array_pop($saher)
?>
```

سيتم حذف العنصر hajeer من المصفوفة ولن يكون في المصفوفة غير ثلاثة عناصر.

Implode و Explode

تقوم هاتين الدالتين باقتصاص قيمة معينة من مصفوفة أو نصوص وتقوم بإضافة قيمة معينة على مصفوفة أو نصوص.

الدالة Implode

تقوم بإضافة قيمة على عنصر بين عناصر المصفوفة.

مثال:

```
<?
 "alfarsi"); ، "ali"، "salem"،$stng =array ("ahmed"
$stng);،$r =implode ("H"
echo $r;
?>
```

الدالة explode

تقوم بحذف قيمة من مصفوفة وذلك لا يعني حذف عناصر من المصفوفة.

مثال:

```
<?
 "alfarsi"); ، "ali"، "salem"،$stng =array ("ahmed"
```

```
$stng);،$r =implode ("-"
echo $r;
$stng);،$r= explode ("-"
echo $r;
?>
```

HTTP_POST_VARS و HTTP_GET_VARS

هذه ليست متغيرات بل مصفوفات، نعم هذه مصفوفات ولكن في ماذا نستخدمها ولماذا؟

في الواقع تحدثنا في الفصل السابق عن طريقة التعامل مع النماذج والحصول على البيانات من المستخدم وتكلمنا عن أسلوبين لنقل البيانات وهما GET و POST عندما تصل البيانات محفوظة في متغيرات إلى صفحة الـPHP فإنه يقوم بتعريفها تلقائياً ويمكنك طباعة المتغيرات وقيمها مباشرة من غير تعريف.... ولكن هذه الميزة في الـPHP يمكن إلغاؤها عن طريق الملف PHP.INI وذلك بإغلاق ميزة register_globals

وذلك بوضع off بدلاً من on

الوضع الافتراضي لها هو on ولكن تستطيع إغلاقها وقد تكون مستأجراً عند مزود خدمة ويب وسيط فيقوم بإغلاق هذه الميزة من باب الحماية ليس إلا.... لا تقلق يمكنك الحصول على البيانات فهي ما زالت موجودة ولكن يجب عليك أن تقوم باستخدام هاتين المصفوفتين لكي تستخرج البيانات.

لنفرض أنك اشتركت عند مزود ويب وكان قد أغلق ميزة (register_globals)

حسناً لنفرض أنك قد صنعت نموذجاً يستخدم مربع نص ويحفظ قيمته في متغير اسمه Dorrah ثم بعد ذلك يقوم بإرسال هذه القيمة باستخدام الأسلوب GET إذا سيكون جزء من الكود في الصفحة الأولى والتي تحتوي على النموذج كالتالي:

```
<form method =get action = “try.php”>
ماهو اسم الطفل الذي استيقظ به العالم الاسلامي من غفلته قبل عدة شهور !!
<br>
<input type=text name = “Dorrah”>
<br>
```

في الملف الثاني(try.php) سنقوم بكتابة الجزء الذي سيقوم بطباعة القيمة كالتالي:

```
<?
Echo HTTP_GET_VARS[“Dorrah”];
?>
```

لاحظ أننا لم نستخدم $ ولكن إذا أردنا الاحتفاظ بقيمة المتغير في متغير آخر فيمكننا ذلك بشكل عادي كالتالي:

```
<?
$Dorrah= HTTP_GET_VARS[“Dorrah”];
?>
```

طريقه بسيطة.... أليس كذلك ولكن.... لنفترض أن مزود خدمة الويب لديك حريص جداً ولذلك فقد ألغى أيضاً ميزة استقبال هذه القيم في المصفوفات.... يمكنه ذلك في ملف الـphp.ini في إعدادات الـtrack_vars الذي يقوم بمنع السيرفر من استخدام هذه المصفوفات (هذه الميزة يمكن إلغاؤها في php4).... على ذلك انصحك بإرسال رسال تذمر وشكوى إلى مزود الخدمة لديك.. تعلن فيها أن الأمر أصبح لا يحتمل.

مصفوفة متعددة الأبعاد

يمكنك صناعة مصفوفات بداخل مصفوفات على حسب ما تحتاجه في معلوماتك الرياضية فقد تحتاج مثلاً إلى إنشاء أشياء معقدة (ومقلقة نفسياً) نريد أن نخبرك على أية حال أنه يمكنك صناعة المصفوفات المتعددة الأبعاد ويمكنك استخدام حتى مئة مصفوفة متداخلة ولكن يجب أن تراعي حجم الذاكرة المستخدمة في السيرفر

لديك (وعلى كل حال إن استطعت أن تقوم بالتركيز في صناعة عشر مصفوفات متداخلة بدون أي مشاكل أو مرض نفسي أو.... فأنت تستحق جائزة).

يمكننا كتابة مصفوفة متداخلة كالتالي:

```
<?
2 => array ("salem ،154786)،$mon= array (1=>array ("sharkeh al-jafali"
1257)); ،almazen"
while (list($personnum) =each ($mon))
{
echo ("<br>$personnum<br>");

$phone)=each ($mon[$personnum]))،while (list(
{
echo ("$phone");
}
}
?>
```

الشرح

هذا المثال قد يكون غامضاً جداً لكن فكرته بسيطة أولاً افترض أنك تعلم عن list..each جيداً وتعرف صيغة التكرار الذي يستخدمهما.

الآن لدينا مصفوفة تتكون من رقمين للفهرسة هذين الرقمين كل واحد منهما عنصره عبارة عن مصفوفة هذه المصفوفة تحتوي على عنصرين (ولنتناسى أنهما يحتويان على أرقام فهرسة) وهما اسم شخص ورقم هاتفه.

في أول خطوة:

```
while (list($personnum) =each ($mon))
{
echo (“<br>$personnum”);
```

قمنا بإخراج رقم الفهرسة الأساسي للمصفوفة والذي يعتبر هو الرقم التسلسلي للأشخاص أصحاب الهواتف ومن بعد ذلك نقوم بطباعة هذا الرقم التسلسلي ويبدأ من سطر جديد.

في الخطوة الثانية:

```
$phone)=each ($mon[$personnum]))،while (list(
{
echo ("$phone");
}
```

نقوم بإخبار الـPHP بطباعة العناصر الذي تحتويها المصفوفة التي تمت طباعة رقم فهرستها ، ولاحظ) ،($phone أنها تشير إلى عناصر مصفوفة وليس فهارسها لأننا تجاهلنا فهارس المصفوفة الداخلية.

لا تقلق الأمر سهل ولكنه يحتاج إلى تدرب فقط، وعليك أن تتدرب وصدقني أنني حاولت أن أبسط المثال من أجلك... أتمنى أن تكون قد فهمت.

تطبيق عملي

افتح محرر النصوص لديك واكتب الكود التالي:

```
<?
Echo "<form method =post action = 'exam2.php' " ;
$boy=array ("أحمد"، " خالد"، " سعد"، "حسن");
$Name) = each ($boy))،while (list(
{
echo " ماهي السنة الدراسية لـ $Name؟";
Echo "<select name = 'school[]'>
<option>أول ثانوي</option>
<option>ثاني ثانوي</option>
<option>ثالث ثانوي</option>
</select>";
echo "<br><br>";
```

```
echo "<input type =hidden name =boy[] value ='$Name'>";
}
echo "<input type =submit ></form>";
?>
```

احفظ الكود باسم exam.php

افتح محرر النصوص واكتب الكود التالي واحفظه في ملف باسم exam2.php

```
<html dir = "rtl">
<?
$V)=each($school))،While (list($I
{
$friendschool[] = $school[$I].$boy[$I];
}
asort ($friendschool);
$V)=each($friendschool))،While (list ($I
{
echo "<br>$boy[$I]"."  ".$school[$I];
}
?>
```

قم بتشغيله بعد نقله لمجلد السيرفر.

الشرح

الذي قمنا به في المثال السابق هو أننا قمنا بإنشاء مصفوفة لعدة أشخاص ($boy) ونريد أن نعرف مرحلهم الدراسية في الثانوية فأنشأنا لكل طالب قائمة منسدلة بواسطة التكرار (list-each) بصناعة قوائم منسدلة وحقول مخفية يتم تخزين قيم الحقول (التي تحتوي على أسماء الأشخاص) في المصفوفة ($boy) وسيتم تخزين نتائج كل القوائم في مصفوفة ($school) وبعد أن يختار المستخدم الإجابات التي تناسبه ويرسل البيانات سيتم استقبال المصفوفة التي فيها نتائج القوائم المنسدلة ($school) واستقبال المصفوفة التي فيها أسماء الأشخاص ($boy) ومن ثم يتم إنشاء مصفوفة جديدة باسم $friendschool[] ويؤخذ

منها معلومات المصفوفتين ويتم دمجها فيها ومن ثم يتم بتكرار آخر طباعة عناصر المصفوفتين $boy و$school.

تكرار **foreach**

هذا التكرار هو من الأشياء الجديدة في الـphp4 وهو يساعدك على معرفة عناصر مصفوفة معينة أو طباعة محتوياتها.

```
Foreach ($ArrayName As $ArrayItem)
{
code شيفره
}
```

مثال:

```
<?
 c=>"car")، b => "basem"، $T= array (a=>"ahmed "

Foreach ($T As $A => $r)
{
echo $A."-----". $r;
}
?>
```

الدالة **count**

تقوم بحساب عدد العناصر الموجودة في المصفوفة

مثال:

```
<?
"c");،"b"،$c=array("a"
$v=count($c);
echo $v;
?>
```

الفصل الخامس
ترتيب الكود البرمجي

تعلمنا في الفصول السابقة أساسيات من أساسيات البرمجة وأعطينا مثال عن الروتين في الحياة اليومية وهو أن تقوم بعمل شيء أكثر من مرة في الحياة اليومية مثل شرب الشاي أو شرب القهوة وغير ذلك، فصلنا هذا يتكلم عن ترتيب الكود ويتكلم تقريباً عن نفس فكرة الروتين اليومي فأنت في حياتك تكرر بعض الأعمال بشكل روتيني

وقد تكون مللت الروتين فأحضرت شيئاً يساعدك على التخفيف من هذا الروتين... فمثلاً عند استخدامك لبرنامج MS Word قد تكون مللت من تنسيق عدة نصوص بطريقة معينة فأنت عند ذلك تقوم بصناعة ماكرو يقوم بفعل العمل الذي كنت تفعله في عدة خطوات بخطوة واحد فقط !!

ولنقل أنك في حياتك اليومية وفي يوم إجازة وقررت أن تقوم بعمل تنظيف شامل (يا إلهي عليك غسيل أطباق الصحون وتنظيف الأثاث وتنظيف الأرضية وترتيب المكتبة وترتيب غرفة النوم و... و... إلخ) عند ذلك فإنك تبحث عن طريقة عملية لكي يتم إنجاز هذه المهمة في أسرع وقت فتقوم بتقسيم هذه المهمة الكبيرة على عده أقسام (التنظيف، الترتيب، الغسيل، .....) ثم تقوم باستدعاء أطفالك وفلذات أكبادك وتقسم على كل واحد منهم مهمة بسيطة يستطيع القيام بها.. هذا التقسيم يسمي في عالم البرمجة بالـfunction (دالة أو وظيفة)

Function

الدالة هي جزء من كود البرنامج يتم تعريفه عن طريق المبرمج ليتم تنفيذ شيء معين بواسطتها، تقوم الدالة بأخذ قيم وتسمى (arguments معطيات) كمدخلات، ثم تقوم بعمل بعض التعديلات على هذه المدخلات وتقوم بإخراج قيمة أخرى في أكثر الأحيان تقوم الدالة بأخذ القيم ووضعها في متغيرات أخرى تسمى بالـ(parameters) لكي يتم إجراء العمليات عليها داخل الدالة وهذه المتغيرات لا

تعمل خارج الدالة أي أنها متغيرات خاصة بالدالة فقط!... في فصولنا السابقة قمنا باستخدام دوال عديدة مثل دوال فرز المصفوفات ودوال إيجاد نوع البيانات، هذه المرة سنقوم ببناء دوالنا الخاصة بنا، ومن صنعنا نقوم بإعطائها المعلومات والبيانات وهي تقوم بإجراء العمليات عليها ومن ثم إخراج الحلول...

تعريف واستدعاء الدوال

لكي تقوم بتعريف دالة فإنك تقوم بكتابة الكلمة function متبوعة باسم الدالة والبارمترات اللازمة والتي سيتم إجراء العمليات عليها بين قوسين ومن ثم تقوم بكتابة الكود اللازم وسط } و {

الصيغه:

```
Function functionname (parameters)
{
function code
}
```

تقوم بكتابه اسم الدالة بدلاً من functionname ثم تقوم بتعريف المتحولات أو المتغيرات parameters ومن ثم تقوم بكتابة الكود الذي سوف يقوم بالمطلوب بين القوسين بدلاً من function code

دعنا الآن نقوم بكتابة دالة من إنشائنا والتي تقوم بإجراء عملية الجمع على متغيرين وسنقوم بتسمية الدالة باسم sumnoraml وهو اسم من تأليفنا ويدل على وظيفة وهدف الدالة ويمكن أن تقوم بتمسية الدالة بأي اسم تريده ولست مجبراً بكتابة اسم معين.

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return $a;
}
?>
```

نقوم في هذه الدالة بإجراء عملية إضافة ١٠٠ على المتغير أو القيمة التي يتم تمريرها.

Return

يجب أن نضعها في نهاية كل دالة، نستخدم هذه الكلمة لكي نقوم بإعلام الدالة أن وظيفتها انتهت وأيضاً نستخدمها إذا كان لدينا أكثر من قيمة ونريد أن نقوم بإخبار الـPHP ما هي القيمة التي سيتم اعتمادها ففي مثالنا هذا أردنا إخبار الـPHP بأن يقوم بأخذ المتغير $a بأنه هو القيمة النهائية مع أنه لو لم نضع المتغير فسيتم اعتباره هو الناتج النهائي لأنه لا يوجد متغير آخر تمت عليه أي عمليات.

الذي أقصده أننا لو كتبنا الكود بالشكل التالي:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
?>
```

فإنه لاضرر من ذلك لأنه لا يوجد لدينا إلا قيمة واحدة لن يتم اعتماد قيمة غيرها ولكن لو افترضنا أنه لدينا أكثر من قيمة كما في المثال التالي:

```
<?
$b)،Function sul($a
{
$a = $a + 100 ;
$b= $b*100;
return $a ;
}
?>
```

هنا يجب تحديد أي المتغيرين سيكون هو القيمة النهائية للدالة.

شرح الدالة (**sumnormal**)

تقوم الدالة التي صنعناها بأخذ قيمتين ومن ثم فإنها تقوم بزيادة العدد الذي يتم تمريره ١٠٠

ولكي نقوم بإخراج نتيجة الدالة فإننا ببساطة نستطيع ذلك بإجراء أحد الأمرين echo أو print.

مثال:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
echo sumnormal(500);
?>
```

لقد قمنا بتمرير رقم بدلاً من المتغير ويمكننا أيضاً تمرير متغير بدلاً من الرقم

مثال:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
$f=100;
echo sumnormal($f);
?>
```

لاحظ أننا استخدمنا متغير في الدالة (مما يثبت كلامنا في الأعلى أن للدالة متغيرات خاصة بها) وليس معنى ذلك أننا لا نستطيع استخدام متغيرات بنفس الاسم المذكور في الدالة فيمكننا مثلاً كتابة نفس اسم المتغير بدون حصول أي مشاكل كالتالي:

```
<?
Function sumnormal($a)
{
```

```
$a = $a + 100 ;
return ;
}
$a=100;
echo sumnormal($a);
?>
```

يمكننا أيضاً استدعاء دالة بشكل عادي إذا كانت هي تقوم بالطباعة

مثال:

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
print $a;
return ;
}

$a=100;
sumnormal($a);
?>
```

print

يقوم الأمر print بنفس عمل الدالة echo ولا يوجد بينهما اختلاف سوى أن الدالة echo قديمة وهي الأصل أما الدالة print فقد تم إنشاؤها في php4 ولا يوجد أي فرق بينهما إطلاقاً.

مثال:

```
<?
Print "أحمد";
?>
```

ويمكننا بها إخراج نتيجة دالة

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
```

```
}
$a=100;

print  sumnormal($a);
?>
```

اين يتم وضع الداله ؟

يمكنك وضع الدالة في أول الكود أو في آخره أي أنه لا فرق بين:

```
<?
//لاحظ أننا قمنا بتعريف الدالة أولاً ثم استدعائها
Function fares($d)
{
print "alfareees@hotmail.com";
}

; fares($d)
?>
```

وبين:

```
<?
//لاحظ أننا قمنا باستدعاء الدالة أولاً ثم تعريفها
; fares($d)

Function fares($d)
{
print "alfareees@hotmail.com";
}
?>
```

يمكنك أيضاً عدم وضع متغيرات في الدالة كالتالي:

```
Html_header ()
{
Print "<html><head><title>alfareees</title></head>";
Return ;
}
```

هذه الدالة تقوم بكتابة الطور الأول من صفحة html لاحظ أننا لم نقم بوضع أي متغيرات أو عوامل أو متحولات (سمها كما شئت).

تمرير القيم إلى الدالة:

هناك نوعين من تمرير القيم

١ - تمرير القيمة مباشرة إلى الدالة (passing by value)

وذلك أن نضع القيمة مباشرة بدون إدراجها في متغيرات.

مثال:

```
<?
Function alfars ($f)
{
$f=$f+$f;
return ;
}
echo alfars(100);
 ?>
```

لاحظ أننا قمنا بإدراج القيمة مباشرة للدالة من غير وضعها في متغيرات.

٢ - تمرير القيمة عن طريق المرجع (passing by reference)

نقصد بهذا أننا نقوم بوضع القمية في متغير أولاً ثم نضع هذا المتغير في الدالة لكي يتم إجراء العمليات عليه مثال:

```
<?
Function alfars ($f)
{
$f=$f+$f;
return ;
}
$r =1000;
echo alfars($r);
 ?>
```

إعداد قيمة افتراضيه للدالة

تستطيع أن تجعل الـPHP4 يقوم بإدراج قيمة افتراضية عند عدم تمرير متغيرات إليه

مثال:

```
<?
Function alfars ($f=40)
{
$f=$f+$f;
return ;
}
echo alfars();
?>
```

إذا لم يتم إعطاء قيمة للدالة فإنها ستفترض أن القيمة هي ٤٠ مباشرة.

أما إذا تم تمرير قيمة أو متغير فإنه سيتم العمل بالقيمة التي تم تمريرها بدلاً من القيمة الافتراضية

مثال:

```
<?
Function alfars ($f=40)
{
$f=$f+$f;
return ;
}
echo alfars(100);
?>
```

مدى المتغيرات [variable scope]

هناك متغيرات محلية (local) ومتغيرات عامة (global)، نقصد بالمتغيرات المحلية التي تكون في داخل الدالة ونقصد بالعامة التي تكون في كود الـPHP بشكل عام.

مثال

```
<?
//هذا متغير عام
$r= "salem";
function ala($s)
{
//هذا متغير محلي
$s = "progrramer";
}
echo $r ;
ala($s);
echo $s;
?>
```

مثال:

```
<?
//هذا متغير عام
$r= "salem";
function ala($s)
{
//هذا متغير محلي
$s = "progrramer";
}
echo $r ;
$s=10;
echo $s;
?>
```

في المثال الأول استطعنا طباعة المتغير $r ولم نسطتع طباعة المتغير $s لأنه محلى (لا يتم تنفيذه الا داخل الدالة) وعندما نريد طباعته فإننا يجب أن نطبع ناتج الدالة لكي نحصل عليه (أي أننا لا نستطيع طباعته بشكل مباشر)

مثال:

```
<?
//هذا متغير عام
$r = "salem";
function ala($s)
```

```
{
//هذا متغير محلي
$s = “programmer”;
}
//استطعنا طباعته بشكل مباشر
echo $r ;
ala($s);
//يجب استخدام الداله لكي تتم طباعته
echo ala($s);
?>
```

لاحظ أننا حتى لو قمنا بعملية طباعة المتغير من نفس الدالة فالناتج يكون مختلفاً لأن لكل متغير عالمه الخاص به

لكي نقوم بجعل المتغير الذي بداخل الدالة متغيراً عاماً فيمكننا ذلك بإحدي الطريقتين التاليتين:

الطريقة الأولى:

```
<?
function ala($y)
{
echo $y. “<br>”;
global $s;
$s = "programmer";
return ;
}
$f =10;
ala($f);
echo $s;
?>
```

لاحظ أننا عندما استخدمنا global في داخل الدالة لكي يتم تعريف أن المتغير متغير عام وبعدما قمنا باستخدام الدالة قامت بطباعة المتغير المراد طباعته ومن ثم بعد ذلك قامت بتعريف متغير جديد ($s) وهذا المتغير متغير عام لأننا وضعنا قبله الكلمة global فاستطعنا طباعته بكل سهولة.

الطريقة الثانية:

هي أن نستخدم المصفوفة $GLOBALS التي تستخدم في PHP لتعريف المتغيرات العامة أيضاً

مثال:

```
<?
function ala($y)
        {
echo $y. “<br>”;
$GLOBALS[“s”] ;
$s = "programmer";
return ;
}
$f =10;
ala($f);
echo $s;
?>
```

المتغيرات المستقرة [static variable]

أقصد بالمتغيرات المستقرة هي التي تكون قيمتها ثابتة

مثال:

```
<?
Function  addfares($y)
{
$y;
$y=$y+1 ;
return $y;
}
echo addfares($y);
echo addfares($y);
echo addfares($y);
echo addfares($y);
?>
```

```
<?
Function  addfares($y)
{
static $y;
$y=$y+1 ;
return $y;
}
echo addfares($y);
echo addfares($y);
echo addfares($y);
echo addfares($y);
?>
```

لاحظ عندما عرفنا المتغير بأنه static فإنه يحتفظ بقيمته حتى لو انتهت الدالة.

دوال متداخلة

يمكننا عمل تعشيش للدوال مثلما كنا نفعل مع بناء القرارات والتكرارات

مثال:

```
<?
Function sum($sa)
{
   $sa=$sa-1;
function goadd ($r)
{
$r = $r+$r;
return $r;
}
$sa= goadd ($sa);
return $sa;
}
echo sum (15);
?>
```

في مثالنا هذا لدينا دالتين الدالة الأولى هي sum والدالة الثانية هي goadd

وظيفة الدالة الأولى هي أن تقوم بالإنقاص من العدد الذي يمرر إليها واحد ثم تقوم بتطبيق دالة داخلية فيها هي goadd تقوم بزيادة العدد على نفسه.. ومن ثم قمنا بنداء الدالة الأولى (لأنها هي الأساس الذي يوجد به الدوال الداخلية) وطباعة قيمتها.

اشتمال الملفات (include files)

قد يكون لديك في برنامجك متغير متكرر في أكثر من صفحة أو رسالة خطأ معينة أو تريد إدرج نص كبير الحجم في صفحات متعددة.
هنا يمكنك اشتمال ملفات في داخل ملفات الـPHP. هذه الملفات قد تحتوي على نصوص أو كود html أو كود PHP.

إن الصيغة التي تستخدمها لاشتمال الملفات هي:

```
Include (filename);
```

مثال:

قم بفتح ملف نصي واكتب فيه ما تشاء ثم احفظه باسم a.txt
قم بإنشاء ملف php واكتب فيه ومن ثم احفظه باسم b.php

```
<?
Include ("a.txt");
?>
```

انقلهما إلى مجلد السيرفر.. شغل ملف الـb.php وانظر النتيجة.
يمكنك أن تقوم بإنشاء ملف PHP وتحتفظ فيه بجميع الـfunction المطلوبة لبرنامجك وعند إرادتك لاستخدام أي واحدة منها تقوم فقط باشتمال الملف ومن ثم استدعائها.

داله تلوين الكود:

هل رأيت مواقع تقوم بتلوين الكود بشكل مذهل مثل موقع zend ؟.... الأمر بسيط كل ما عليك أولاً

قم بوضع الكود في ملف نصي وسمه بأي اسم (مثلاً file.txt) وبعد ذلك قم باستخدام الدالة

Show_source

مثال:

```
<?
show_source ("file.txt");
?>
```

الفصل السادس
تتبع وتصيد ومنع الأخطاء
(avoiding and handling errors)

إن مصطلح الـdebbug هو من المصطلحات الشائعة والشيقة في عالم البرمجة، هذا المصطلح يشير إلى كيفية إصلاح أخطاء البرنامج وتوقعها قبل حدوثها، هناك أنواع من الأخطاء تحدث بسبب المبرمج وهناك أنواع من الأخطاء تحصل بسبب المستخدم، في العادة يجب أن يكون المبرمج متآلفاً مع مصطلح تتبع الأخطاء وإصلاحها.

قد يكون من أهداف تتبع الأخطاء الحماية بقدر أهمية البرنامج الجاري العمل عليه أو الموقع فكلما كان الموقع مهماً كان وجوب حمايته أكبر.

قد يكون من الأسباب التي تسبب تدميراً للمواقع هو أن صاحب الموقع يغطي كل صغيرة وكبيرة عن برنامجه الذي يركبه في موقعه وقد يكون برنامجه هذا غير محمي بشكل كاف أو يكون مسير بعدة ملفات فيقوم شخص بحذف ملف من الملفات الأساسية بسبب عدم دقة في التراخيص المعطاة مما يؤدي إلى دمار الموقع نهائياً.

وقد يكون صاحب الموقع مهملاً في الحد ذاته فلا يحتفظ بالمعلومات السرية لموقعه مما يسبب مشاكل أكبر من التدمير مثل احتلال الموقع بشكل كامل.

رسائل الخطأ في الـPHP لها طريقتها وتقنيتها الخاصة التي تسير عليها فهي ليست مثل الجافا وليست مثل cgi
فالـ PHP لا تقوم بإرسال الخطأ إلى السيرفر بل تقوم بكتابة رسالة خطأ في مكان الخطأ.

قد يكون هناك أخطاء يصعب تتبعها أو معرفة مكانها في الأصل، وقد يكون هذا بسبب أنك تستخدم الـPHP في صناعة موقع ديناميكي وتشرك معها الجافا سكريبت وتضع علامات التعليق الخاصة التي تقوم بإخفاء الأخطاء في الجافا مما قد يجعلك تشعر بالحيرة وتجن أين مكان الخطأ.

```
<!- -
رساله الخطأ
- - >
```

أنواع الأخطاء

هناك أنواع من الأخطاء منها الإملائية (Syntax Error) ومنها المنطقية ومنها أخطاء تحدث في وقت التنفيذ

ومثال الأخطاء الإملائية:

```
<?
Eco "1";
//من المفترض أن تكت التالي:
Echo "1";
?>
```

هذا سيعطيك رسالة خطأ Parse error

ومن الأخطاء الإملائية نسيان الفاصلة المنقوطة (semi-colon) في نهاية الدالة:

```
<?
Echo "hello"
//من المفترض أن تكتب التالي:
Echo "hello";
?>
```

هنا سوف يعطيك الـPHP رسالة خطأ لكن العجيب أنه لن يعطيك إياها بشكل صحيح فرسالة الخطأ تشير إلى أن السطر الرابع يحتوي على الخطأ بينما الخطأ هو في السطر الثاني.

وهناك خطأ آخر يحصل بسبب نسيان الـbrace (وهي الاقواس):

```
<? Php
for ($loop = 0 ; $loop < 5 ; $loop ++)
{
Echo “”;
?>
```

إذا كنت قد نسيت إغلاق القوس فهذا من الأخطاء الشائعة، والأخطاء الإملائية لايمكن حصرها، إنها أشبه بقواعد اللغة، لكن أكثر الأخطاء الإملائية الشائعة في برامج الـPHP

١ – نسيان الأقواس.

مثال:

```
<?
for ($loop = 0 ; $loop < 5 ; $loop ++)
{
for ($loop1 = 0 ; $loop1 < 10 ; $loop1 ++)
{
for ($loop = 0 ; $loop < 5 ; $loop ++)
{
code ....
}
}
```

في المثال السابق ينقصنا قوس إغلاق التكرار الأخير ({)

٢ - نسيان الفاصلة المنقوطة.

مثال:

```
<?
Echo 10
<?
```

٣ - خطأ إملائي في اسم **function**.

مثال:

```
<?
Htmlspecialchar($I);
?>
```

سيعطيك رسالة خطأ:

```
Fatal error: call to Undefined function: htmlspecialchar().
```

وتصحيحها أن تكون:

```
<?
Htmlspecialchars($I);
?>
```

٤ - نسيان إغلاق النص.

مثال:

```
<?
Echo “arabbuilder;
?>
```

نسي الـ(")في نهاية الكلمة. وسيعطيك Parse error

الأخطاء المنطقية [Logical Errors]

إن الأخطاء المنطقية هي الأكثر صعوبة في التتبع فقد تجد برنامجك يعمل بشكل صحيح وبكل سلامة ولكنه عند نقطة ما لا يتم تنفيذها كما تريد أنت، لنضرب مثلاً على خطأ منطقي بسيط جداً، لنفرض أنك قمت بعمل نموذج مكون من مربع نص وزر، عند ضغطك لهذا الزر فأنت تريد أن تتم كتابة كلمة كبير إذا كان الرقم أكبر من ٣٠ وكلمة صغير إذا كان الرقم أصغر من ٣٠ لنقم بكتابة الكود للمثال الأول:

```
<?
echo "أدخل عمرك:";
echo '<br>
<form method = "post" action = "age.php">
<input type= "text" name = "age">
<br>
<input type= submit value = "هل أنا كبير أم صغير؟" >
</form>' ;
?>
```

في ملف age.php اكتب الكود التالي:

```
<?
If ($age<30) echo “أنت صغير”;
If ($age>30) echo “أنت كبير”;
?>
```

سيعمل السيكريبت بشكل صحيح.. ولكن ربما تخطئ أنت في كتابة العلامات المنطقية (التي باللون الأحمر) فتأتي النتائج بشكل خاطئ.

ومن الأخطاء المنطقية الأخطاء التي تقع في وقت التشغيل (Run times error) والتي قد تقوم بإيقاف برنامجك بشكل كامل

مثال:

```
<?
$t=0;
$r=1;
$f=$r/$t;
?>
```

وعندها سينتج لك الرسالة التالية

Warning: Division by zero in (path) on line (line number)

هناك نوع آخر من الأخطاء المنطقية (unexpected) وهو لا يقوم بإيقاف البرنامج نهائياً بل يقوم بإخراج رسالة الخطأ في مكان الخطأ أو قد يقوم بتنفيذ البرنامج وإخراج البيانات بشكل غير صحيح أو قد لا يقوم بإخراج بيانات
وهو المثال الأول الذي ذكرناه سابقاً (تقييم العمر).

أخطاء التكرارات:

قد يكون لديك أيضاً تكرار فيه خطأ ولا يقوم بالتوقف نهائياً مثل هذا التكرار:

```
$c=1;
$t=true;
while ($t=true)
{
$c++;
}
```

لم نقم بعمل شي يوقف التكرار مثل أن تضع شرط يختبر قيمة المتغير ($c) ثم يقوم بإيقافه عند تعديه رقم معين وعلى ذلك فإن التكرار سيستمر بشكل غير متوقف ولن يعمل البرنامج.

عدم إرجاع قيمه من function:

مثال:

```
<?
Function ($d)
{
$d =$d+$d;
}
```

الخطأ هنا أننا لم نستخدم الـreturn لكي ننهي الدالة أو قد تكون الدالة تحتوي على أكثر من قيمة وننسى أن نقوم بتحديد القيمة النهائية للدالة.

الخلط في المعاملات الحسابية والمنطقية:

مثال:

```
If ($y=10) echo 12 ;
```

والمفترض أن تكون:

```
If ($y= =10) echo 12 ;
```

أفكار جيدة لتفادي الأخطاء:

التعليقات:

إن من الأفكار الجيدة للتقليل من الأماكن التي تبحث فيها عن الخطأ هو وضع تعليقات لوصف وظيفة دالة معينة.

مثال:

```
<?
//هذه الكود يقوم بطباعة كلمة أحمد
" ; أحمدEcho "
?>
```

الدوال

وأيضاً من الأفكار الجيدة أن تقوم بتقسيم وظائف البرنامج على دوال بحيث أن لكل دالة وظيفتها المعينة:

```
<?
/*
+- -----------------------------------------------------------+
 |              هذه الدالة تقوم بقسمة العدد على ٢              |
+- -----------------------------------------------------------+
* /
function ($U)
{
$U=$U/2;
return $U ;
}
?>
```

Regular Expressions

هـذه التقنيـة تـساعدك علـى تفـادي الأخطـاء في صـفحتك عنـد حـدوثها مثـل أن يقـوم مستخدم ما بكتابة بريد إلكتروني غير صحيح (مثال: a@y@.k.d) هـذا البريـد غيـر صـحيح ولأجـل أن تقـوم بمنـع حـصول أي خطـأ مثـل ذلـك وتقييـد العبـارات الـتي يدخلها المستخدم فإنك تقوم باستخدام الـ/RE (Regular Expressions) إنـك بالأصح تجعل قواعد للكلمات التي يدخلها المستخدم فمثلاً تجعل المستخدم لا يدخل سـوى أرقـام أو حـروف فقـط أو شـكل معـين مـن الكلمـات، تقـوم أولاً بإنـشاء نمـط للكلمة التي تريد المستخدم أن يقوم بإدخالها.

النمط [pattern]

مـاهو الـنمط؟ مـا رأيـك إذا كتـب المـستخدم جملـة في مربـع نـص تحتـوي علـى عـدة كلمات وتريد أن تتأكـد من وجود كلمة معينـة وسـط هـذه الجملـة، علـى حـسب مـا أخذناه من معلومات عن المصفوفات سابقاً نستطيع فعل ذلك كالتالي:

```
<?
";، five، four، three، two،$words="one
$ty);،$ty =explode (""
foreach ($ty as $w) {
 "six") echo "found string 'two'"; if ($w = =
}
?>
```

لقد كـان المتغير $words يحتوي على جملة تتكون من عدة كلمات وعندما أردنـا فحصه قمنا باستخلاصه في مصفوفة ثم بعد ذلك قمنا بفحص المصفوفة باستخدام التكـرار foreach، ومع ذلك فإن الذي فعلناه بهذا الاستخدام غير عملي بتاتاً وهنـا تـبرز قـوة Regular Expressions لاحظ الآن كيـف نـستخرجه بواسـطة الـ Regular Expressions:

```
<?
";، five، four، three، two،$words="one
 ‘ one’ “ ;  لقد وجدت العدد$words)) echo “،if (ereg(“one”
?>
```

في هـذا المثـال قمنـا باسـتخدام الدالـة (ereg) ووضـعنا في خانتهـا الأولى الـنمط (pattern) الـذي نريـد أن نتأكـد مـن وجـوده (أوالكلمـة المـراد البحـث عنهـا) ووضعنا في الخانة الثانية المتغير الذي سيتم البحث فيه عن الكلمة أو النمط.

تقوم الدالة ereg بإعطاء القيمة true إذا تم العثور على الكلمة.

في الواقع هناك استخدامات أكثر فعالية للأنماط.

يمكننا مثلاً تخزين الكلمة إذا تم وجودها في مصفوفة خاصة كالتالي:

```
<?
";، five، four، one، two،$words="one
$rok))  ;،$words،if (ereg(“one”
echo $rok[0];
echo $rok[1];
?>
```

نقوم بوضع اسم المصفوفة حيث نريد تخزين البيانات في الخانة الثالثة.. لاحظ مع أنه يوجـد كلمـتين في الجملـة توافـق الـنمط إلا أنـه أعطانـا كلمـة واحـدة فقـط إذ إن وظيفته أن يتأكد من وجود النمط في الجملة فقط فإذا تأكد من وجوده مرة واحدة استكفى واعتبر الموضوع قد انتهى.

ماذا لو أردنا أن نتأكد من عدة كلمات، عند ذلك فإننا نفعل التالي:

```
<?
";، five، four، one، two،$words="one
$rok))  echo $rok[0];،$words،if (ereg(“one”
$rok))  echo $rok[0];،$words،if (ereg(“two”
?>
```

وأريد أن أنبهك أن الـereg يقـوم بإنشـاء المصفوفة مـن جديـد عنـد كـل اسـتعمال لـه فخذ حذرك من هذه النقطة

أيضاً فإن الـereg حساس لحالة الأحرف لاحظ هذا المثال:

```
<?
";، five، four، vcx، two،$words="one
$rok))  echo $rok[0];،$words،if (ereg("One"
?>
```

لن يقوم بإخراج أي شي فقط لأن حرف الـO مختلف.

أيضاً يمكنك البحث عن كلمة يسبقها فراغ مثلاً كالتالي:

```
<?
";، five، four، vcxone، two،$words="one
$rok))  echo $rok[0];،$words،if (ereg("one"
?>
```

مثال آخر:

```
<?
";، five، four، vcxone، two،$words="oned
$rok))  echo $rok[0];،$words،if (ereg("one"
?>
```

لاحظ في هذين المثالين أنه مع أن كلمة one غير موجودة بمفردها إنما موجودة كجزء من vcxone وoned ورغم ذلك فإن الدالة لم تأخذ اعتباراً لذلك بينما لو كتبنا كالتالي:

```
<?
";، five، four، vcxone، two،$words="oned
$rok))  echo $rok[0];،$words،if (ereg(" one"
?>
```

فإنه سيبحث عن الكلمة مفصولة عن أي حرف ولن يجد كلمة كذلك فلن يقوم بكتابة أي شي.

يمكننا أن نفحص قيمة موجودة في متغير كالتالي:

```
<?
$reu = "one";
";، five، four، vcxone، two،$words="one
$rok))  echo $rok[0];،$words،if (ereg($reu
?>
```

هل لاحظت أننا فحصنا قيمة المتغير $rue بواسطة ereg مع $word ولم يتطلب منا ذلك أي شيء إضافي غير اسم المتغير المراد البحث عن قيمته في الجملة.

يمكننا بالـ Regular Expression استعمال بعض الأحرف بشكل خاص التي لها استعمالها الخاص بواسطة الـ Regular Expressions

الأحرف الخاصة في الـ Regular Expression هي كالتالي:

```
. * ? + [ ] () { } ^ $ | \
```

هذه الأحرف لها معناها الخاص في الـ Regular Expression

فقديماً مثلاً كنا نقول إنه لا يمكننا أن نستخدم علامتي تنصيص متداخلة من نفس النوع كالتالي:

```
<?
$r="u\"";
?>
```

ولكي يتجاهل الـPHP هذا المعنى فإننا نقوم بوضع (\) قبل علامة التنصيص.

أيضاً مع الـ ereg فإن للـ(.) قداستها ولكي يتم تجاهلها فإننا نستخدم الـ(\)

تقوم الـ(.) بأخذ مكان حرف أو فراغ فمثلاً لاحظ المثال التالي:

```
<?
$P="I love yamen";
$R)) echo $R[0];،$P،if (ereg ("love...."
?>
```

هل لاحظت الناتج ؟؟

ولكي يتم تجاهل قداسة الـ(.) في الـRegular Expressions نقوم بوضع (\) قبلها.

مثال:

```
<?
$P="I love yamen";
$R)) echo $R[0];،$P،if (ereg ("love\.\.\.\."
?>
```

في هذا المثال لن تتم طباعة أي شيء لأنه لا يوجد أي كلمة تطابق (....love) لأن الـ(**.**) فقدت قداستها وبدأ التدقيق في الكلمة حرفاً حرفاً.

صناعة فئة حروف [xyz]

أقصد بذلك أنني أحدد نطاق معين من الكلمة من الممكن أن يكون في هذا النطاق أي حرف من الفئة التي أقوم بتحديدها أو الحروف التي أقوم بتحديدها.

مثال:

```
<?
$y="how are you ? " ;
 $y)) echo "true";،if (ereg("h[oe]"
?>
```

هنا قام الـregular expression بالبحث عن أي كلمة تبدأ بالحرف h ومن ثم يتبعها أحد الحرفين o أو e

مثال هذه الكلمات:

Hey – He – Hew - Homer

ولكنها لاتطابق:

Hty – Hnt - Hlay

أتمنى أن تكون فهمت ما أرمي إليه.

يمكننا أيضا أن نقوم بإخبار الـregular expression بأن لا يقوم باختيار كلمات تحتوي على حروف معينة وذلك فقط بإضافة ^

```
<?
$y=”how are you ? “ ;
 $y)) echo “true”;،if (ereg(“h[^oe]”
?>
```

نقوم هنا بإخبار الـre بأن يقوم بفحص الجملة فإذا وجد أي كلمة تبدأ بـh ولا تحتوي على o أو e فإنه يقوم بإعطاء true وإذا لم يجد يقوم باعطاء false

وهذا الكلام يطابق الكلمات التالية:

Hay - Hana - Hkg

ولا يوافق هذه الكلمات:

Home – Hore - Here

يمكننا استعمال اختصارات لبعض الأمور فمثلاً إذا كنا نريد كلمة لا تحتوي على أي رقم كنا سنكتب كالتالي:

[^123456789]

يمكننا أن نستعمل اختصار لهذا الموضوع كالتالي:

[^0-9]

وحتي إذا أردنا أن يتأكد من وجود رقم من واحد إلى تسعة فقط علينا مسح الـ^

[0-9]

وأيضاً الحروف الصغيرة من a إلى z

[a-z]

وإذا أردنا التأكد من عدم وجودها

[^a-z]

نفس القصة مع الحروف الكبيرة.

هناك اختصارات أخرى لهذا الموضوع كالتالي:

الاختصار	المطابق له	معناه ووظيفته
\d	[0-9]	أي رقم من ٠ إلى ٩
\D	[^0-9]	ممنوع الأرقام من ٠ إلى ٩
\w	[0-9A-Za-z_]	أي رقم من ١-٩ أو حروف A-Z أو أحرف صغيره أو_
\W	[^0-9A-Za-z_]	عكس السابق
\s	[\t\n\r]	يقبل مسافة أو سطر جديد أو علامة جدولة (tab)
\S	[^\t\n\r]	عكس السابق

تحديد مكان الكلمة:

يمكننا أن نقوم بتحديد مكان الكلمة، أقصد بذلك أنه يمكنك تحديد مكان الكلمة إذا كانت في بداية أو نهاية النص ونستخدم لهذا الأمر العلامتين (^) لتحديد المكان لبداية الجملة و ($) لنهاية الجمل.

مثال:

```
<?
$y="how are you ? " ;
$y)) echo "true";،if (ereg("^h"
?>
```

هنا سيقوم الـphp بالبحث عن H في الجملة فإذا وجد الجملة تبدأ بحرف h كانت قيمة الـereg تساوي true وإذا لم يجد كانت قيمة الـereg تساوي false .

```
<?
$y="how gone?" ;
$y)) echo "true";،if (ereg("^g"
?>
```

في هذا المثال ستكون قيمة الـereg خطأ لأن العبارة لا تبدأ بحرف g

يمكننا فعل العكس بواسطة العلامة ($) التي عملها عكس (^) فهي تفحص إذا كان الحرف المراد فحصه موجود في نهاية الجملة.

مثال:

```
<?
$y="how g" ;
$y)) echo "true";،if (ereg("g$"
?>
```

يمكننا أيضا اختيار إذا ما كان واحد من النمطين صحيحاً بواسطة العلامة (|)

```
<?
$y="how g" ;
$y)) echo "true";،if (ereg("^y | g$"
?>
```

في هذا المثال سيقوم الـPHP بفحص الجملة فإذا وافقت أحد النمطين كانت قيمة الـereg عند ذلك true.

يمكننا أيضاً تحديد إذا ما كان حرف أو جملة متكررة بعدد من المرات أو مرة واحدة باستخدام أحد هذه الثلاثة رموز (❖، +، ?)

تقوم علامه الضرب بالتحقق من أن الحرف الذي يسبقها مكرر مرة أو أكثر أو غير موجود بتاتاً

مثال:

```
Bea*t
```

وتوافق:

Bet
Beat
Beaat

تقوم علامة الجمع (+) بالتأكد من وجود عنصر مرة أو أكثر:

```
Bea+t
```

وتوافق:

Beat
Beaat
Beaaaaat

أما علامة الاستفهام فتقوم بالتأكد من وجود عنصر مرة واحدة أو عدم وجوده بتاتاً:

```
Bea?t
```

وتوافق:

Bet
Beat

وتأكد دائماً أن هذه الثلاث علامات مسبوقة بحرف.

وعند إرادتك مثلاً التأكد من سبق حرفين أو ثلاث بشكل تحديدي يمكنك استخدام القوسين

مثال:

```
(wo)?man
```

ويوافق:

man
woman

يمكننا التأكد من تكرر حرف بشكل معين من المرات أو أكبر من عدد معين من المرات أو أصغر من عدد معين من المرات باستخدام القوسين {x، y}

فمثلاً لو أردنا أن نتأكد من أن حرف (d) مكرر مرتين إلى أربع مرات:

```
d{2،4}
```

أما إذا أردنا أن نتأكد من أنه مكرر أكثر من مرتين إلى عدد غير محدود من المرات:

```
d{2،}
```

أما إذا أردناه أن يتكرر ٤ مرات على الأكثر:

```
4}،d{
```

أو إذا أردناه أن يتكرر بعدد محدود من المرات:

```
d{8}
```

أخيراً نريد أن نلفت النظر إلى الاختصار (\b) الذي معناه أي شيء ولكن ليس حرفاً (الحروف التي بين \w وبين \W تقريباً)

ملخص ما أخذناه من القواعد تجدونه في الجدول التالي:

القاعدة	المعنى
[abc]	أي حرف كان a أو b أو c
[^abc]	أي حرف غير a و b و c
[a-z]	كل الحروف من a إلى z
\d\D	\d للأرقام و \D لغير الأرقام
\w\W	\w للحروف جميعها و\W لغير الحروف
\s\S	\s للفـراغ (space) و\S لغـير الفـراغ (no space)
\b	الحروف التي بين \w و\W
.	أي حرف
(abc)	تقوم باعتبار abc كمجموعة..
؟	حرف أو مجموعة حروف مكـررة مـره أو غير مكـررة نهائياً
+	حرف أو مجموعة حـروف تتكـرر مـرة أو أكثر
*	حرف أو مجموعة حـروف تتكـرر مـرة أو

	أكثر أو قد لا تتكرر نهائياً
y}، {x	تكرير بعدد معين من المرات..
y}، {	تكرير بحد أقصى من المرات..
}، {x	تكرير بحد أدنى من المرات...
{x}	تكرير بعدد معين من المرات
^	في بداية النص
$	في نهاية النص

تعبير للتاكد من إيميل

^[_a-zA-Z0-9-]+(\.[_A-Za-z0-9-]+)*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$

شرح التعبير

الرمز	الشرح
^	يجب أن يبدأ النص
[_A-Za-z0-9-]	أي حرف من a-z كبيراً كان أو صغيراً أو أو_ أرقام
+	وقد يكون هذا الحرف متكرراً أكثر من مره
(\.[_A-Za-z0-9-]+)	بالاضافه إلى أنه قد يتبع النقطة وحروف وأرقام
*	وقد لا يتبعه أو قد يتبعه ويتكرر أكثر من مرة
@	وبعد ذلك يكون لديه حرف ال@
[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$	وأيضاً نفس القواعد في النهاية

مثال:

```
<?
$t)،Function mailcheck($mail
{
$T="^[_a-zA-Z0-9-]+(\.[_A-Za-z0-9-]+)*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$";
$mail))،If (EREG($T
{
$r="the mail is true";
echo $r;
}
else
{
$r="the mail is not true";
echo $r;
}
return ;
}
$t);،mailcheck("alfareees@hotmail.com"
?>
```

eregi()

الفـرق بـين هـذه الدالـة والدالـة ereg أنهـا غـير حـساسة لحالـة الأحـرف كـبيرة أو صغيرة أي أنه يمكننا كتابة المثال السابق كالتالي:

```
<?
$t)،Function mailcheck($mail
{
$T="^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*$";
$mail))،If (EREG($T
{
$r="the mail is true";
echo $r;
}
else
{
$r="the mail is not true";
echo $r;
}
return ;
```

```
}
$t);،mailcheck("alfareees@hotmail.com"
?>
```

ereg_replace()

ماذا لو أردت تحرير عبارة ما من أحرف معينة وقد تكون متكررة في جملة أو غير ذلك.

لنفرض أن لدينا العبارة التالية:

```
Mohmed love his game .......
```

ونريد أن نتخلص من النقاط التي في نهاية العبارة

أو لدينا مثلاً هذا المسار:

```
C:\windows\desktop
```

ونريد أن نستبدل العلامة (\) بـ (/)

كل ذلك ممكن بواسطة الدالة ereg_replace وقواعد الـ rgular expression التي أخذناها سابقاً

البنية التي نستخدمها للدالة كالتالي:

```
var);،string،Ereg_replace(reg
```

نضع في مكان reg القاعدة لل regular expression ونضع مكان الـstring الحرف الجديد ونضع بدلاً من الـvar المتغير الذي نريد استخلاص الحروف منه.

مثال:

```
<?
$path =" C:\windows\desktop";
$tell= "Mohmed love his game.....";
$path);،"/"،$newpath= Ereg_replace("[\.]"
$tell);،""،$newtell= Ereg_replace("\."
echo $newpath;
echo "<br><br>";
echo $newtell;
?>
```

أساليب أخرى لتتبع الأخطاء

استخدام عباره echo

هو من أقدم الأساليب وكان يستخدم مثلاً في فحص بعض متغيرات نموذج، فمثلاً أنت لديك نموذج يقوم بإرسال معلومات إلى النموذج وقد تستخدم في اختبار الأخطاء المنطقية التي يستصعب متابعتها في الكود.

مثال:

```
<?
Echo "this is: $name";
Echo "<br>";
Echo "this is: $Email";
//كود يقوم بمعالجة معلومات المتغيرين
//طباعة المتغيرين بعد أداء عملية المعالجة ورؤية النتائج
Echo "this is after: $name";
Echo "<br>";
Echo "this is after: $Email";
?>
```

فحص كود الـ html

قد تستخدم كود جافا سكريبت ويتم إخفاء الأخطاء وسط علامات التعليقات فعليك حينئذ فحص كود الـhtml لرؤية إن كان هناك بعض الأخطاء المخفية أم لا.

تجاهل الأخطاء

لنفترض أنك تعلم أن الدالة التي صنعتها بها أخطاء ولكنك تريد تجاهل هذه الأخطاء فكل ما عليك أن تقوم به وضع @ أمام الدالة لكي يتم تجاهل الخطأ عند حدوثه.

مثلاً نحن نعلم أن القسمة على الصفر من الأشياء الغير مقبولة في الـPHP وأنت صنعت دالة تقوم بالقسمة على صفر ولن يتم تنفيذها لأنها بالأصل خطأ ولكنك تريد أن يقوم PHP بتجاهلها فكل ما عليك أن تفعله هو وضع @ أمام الدالة.

مثال:

```
<?
function amail ($y)
{
$y=$y/0;
return $y;
}
$s= @amail(44);
echo $s;
?>
```

الفصل السابع
التعامل مع العميل

كما رأينا في الفصول السابقة، فإن الـPHP يوفر رقماً عظيماً من المميزات عن الـhtml لبناء مواقع الويب، من الأشياء الأساسية التي لم نتكلم عنها حتى الآن هي الموثوقية (أو الاستقرار) وهو بالمعني الصحيح والصريح:

القابيلة على الاحتفاظ بالمعلومات بين صفحتين منفردتين أو مختلفيتن في المستعرض...

بدون أي إضافات، HTTP لا يوفر أي ميكانيكيه للحفاظ على البيانات وجعلها مستقرة لمعالجة تتم بين صفحتين، كل طلب لصفحة في الإنترنت (request) ليس له أي علاقة بأي طلب آخر... مثلاً عندما تطلب موقع المطور العربي ومن ثم منتدى المطور العربي فإن كلا الطلبين ليس لهما علاقة ببعضهما...

بمصطلح آخر يمكننا أن نقول أن الـHTTP فاقد لحالته (stateless) أي أنه لا يعرف، أي أن أمر طلب الصفحة ينتهي عند انتهاء الطلب، فهي عندما تقوم بنقل بيانات صفحة من السيرفر إلى المستخدم فهي تعرف من هو المستخدم الذي يطلب البيانات وعلى أي نافذة سيتم نقل البيانات وعند انتهاء ذلك فإن كل هذا الموضوع ينتهي وإذا عاد المستخدم فطلب صفحة أخرى فإنه لا يعرف إن كان هو نفس المستخدم أو لا !

إن القدرة على الحفاظ على وجود البيانات ليست وسيلة أو ميزة أو قوة مقتصرة على الـPHP فقط.

فلقد رأيت كيف استطعنا إرسال معلومات من صفحة إلى صفحة بدون خسران أي معلومات وذلك عن طريق الـhtml وبالرغم من ذلك فإن المستخدم عندما يقوم بإغلاق الصفحة عند استقبالها للبيانات فإن ذلك يعني فقدانها للأبد ، عن طريق استخدام الـPHP يمكننا إخبار السيرفر بأن يقوم بإرجاع البيانات بطريقة تمكننا من الحفاظ عليها ، مثلما سنري في هذا الفصل، هناك ثلاث طرق لعمل ذلك....

التميز الحقيقي في قوة الفهم للـPHP، يتطلب منا مفهومية جيدة في كيفية استعمال الـPHP في التفاعل مع المستخدم والمتصفح الذي يستخدمه لكي نتغلب على نقاط الضعف التي في الـhttp.

هذا هو موضوعنا لهذا الفصل والذي سنتكلم فيه عن:

١ - الـHTTP والـ html ومحدودية قدراتهم، وكيف يستطيع الـPHP التغلب على القصور فيهم.
٢ - الاحتفاظ بالمعلومات التي نريد أن نستخدمها بين طلب لصفحتين مختلفتين.
٣ - مكنكة الحفاظ على البيانات.
٤ - الكعكات (cookies) وكيفية استخدامها.
٥ - PHP4 والـ native session – المكنكة الداخلية للحفاظ على وجودية البيانات.

هذا الفصل مفيد بشكل ظاهري لمن هو جديد على إنشاء مواقع متفاعلة متوسطة – كبيرة الحجم بواسطة الـPHP..
إنه يحتوي على الكثير من بعض الأمثلة التي تفيد.

الهدف من هذا الفصل هو أن تتعرف على كيفية الحفاظ على معلومات المستخدم عبر متغير أو أكثر بين أكثر من صفحة، مثل أن تجعل اسم المستخدم ظاهراً في كل صفحة يقوم بالولوج إليها... مما يؤكد استمرارية وجود البيانات.

لنفرض أن لدينا موقعاً على الإنترنت هذا الموقع يهتم ببيع وتسويق مواد غذائية أو أن هذا الموقع يقدم مسابقات ثقافية، في العادة عندما يقوم المستخدم بطلب شراء سلعة معينة أو عندما يختار الدخول في مسابقة من المسابقات الثقافية فإنه يقوم بدخول أكثر من صفحة بالتتابع.

يختار السلعة في الصفحة الأولى وبعد ذلك يقوم برؤية معلومات السلعة في الصفحة الثانية والصفحة الثالثة يقوم فيها بتعبئة معلوماته للشراء أو غير ذلك إلى أن ينتهي من كافة المعلومات وبعد ذلك تنتج له في النهاية صفحة فيها معلوماته والسلعة التي قام باختيارها وفاتورة الشراء !!

أو يقوم باختيار نوع المسابقة الثقافية في الصفحة الأولى وبعد ذلك يقوم بالحصول على عدة أسئلة مقسمة على عدة صفحات إلى أن ينتهي من المسابقة فتخرج له في النهاية مجموع الدرجات للأسئلة ومعلوماته وهل هو فائز أم خاسر!!

في الواقع هذا ما يسمونه بالمحافظة على الجلسة (maintain session) وأقصد بذلك دخول المستخدم إلى صفحة وانتقاله من صفحة إلى صفحة مع المحافظة على معلوماته وغير ذلك من البيانات، لكي نستطيع متابعته أولاً بأول.

في بروتوكول الـhtml والـhttp لا نستطيع معرفة إذا ما كان الشخص عندما يطلب صفحة ما هو نفسه عندما يذهب إلى الصفحة الثانية إذ إن المستخدم عندما يطلب صفحة ما (request) من السيرفر فإن السيرفر يقوم بمعرفة من أي مكان بالعالم يتكلم هذا الشخص ويقوم بإرسال استجابة إليه باعطائه الصفحة التي كان

يطلبها (response) ولكن بعد ذلك فإن السيرفر لا يعرف إذا كان هذا الشخص هو نفسه الذي يقوم بطلب الصفحة الثانية أو الثالثة في السيرفر.

هنا تأتي ميزة الـPHP وغيره من لغات برمجة الإنترنت لصناعة ميكانيكية إبقاء تفاعل مستمر بين المستخدم والسيرفر عن طريق الـsession و الـcookie، ولكي لا نعقد الموضوع دعونا نتكلم عن ذلك عملياً فذلك أفضل لفهم الموضوع من الثرثرة التي لا فائدة منها.

استخدام الحقول المخفية

سنقوم الآن بإنشاء ثلاث صفحات، الصفحة الأولى تطلب من المستخدم إدخال اسمه، والصفحة الثانية تقوم بالترحيب به واعطائه ثلاثة أسئلة، والصفحة الثالثة تقوم باعطائه النتيجة.

افتح محرر نصوص لديك واكتب الكود التالي:

```
<p dir="rtl" align="center">ادخل اسمك الكريم</p>
<form method="POST" action="quiz2.php">
<hr>
<input type="text" name="name" size="20"><br>
<input type="submit" value="إرسال" ></p>
</form>
```

احفظها باسم quiz.php

قم بفتح محرر النصوص واكتب الكود التالي:

```
<html dir ="rtl">
<?
If  (isset($name)) {
Echo "مرحبا بك يا ". $name ;
Echo '
<br>
<form method="POST" action="quiz3.php" dir="rtl">
```

```
<input type="hidden" name = "thename" value =
"'.$name.'">
من هو أول الخلفاء الراشدين ؟</p>
<p dir="rtl"><input type="radio" value="أبو بكر الصديق" name="khlifa">أبو بكر الصديق.</p>
<p dir="rtl"><input type="radio" value="عمر بن الخطاب" checked name="khlifa">عمر بن الخطاب</p>
<p dir="rtl">من هو الفاروق ؟</p>
<p dir="rtl"><input type="radio" name="faroq" value="عمر بن الخطاب">عمر بن الخطاب</p>
<p dir="rtl"><input type="radio" name="faroq" value="سالم بن عامر" checked>سالم بن عامر</p>
<input type ="submit" value = "إرسال" dir="rtl">
</form>' ;
}
else
{
echo "غير مصرح لك بدخول هذه الصفحة" ;
}
?>
```

احفظها باسم quiz2.php

قم بفتح المفكرة واكتب الكود التالي:

```
<?
If  ((isset($thename)) && (isset($khlifa)) &&
(isset($faroq)))
{
echo 'لقد انتهت المسابقه يا '. $thename ;
$range=0;
$co = 0;
```

```
if ($khlifa =="أبو بكر الصديق") {
$range=$range+10;
$co = $co +1;
}
if ($faroq =="عمر بن الخطاب")
{
$range=$range+10;
$co=$co+1;
}
if ($range < 10)
{
echo "ليس هناك أي إجابة صحيحة";
}
else
{
echo "<br>"."عدد الأسئلة التي أجبت عليها =". $co ;
echo "<br>". "الدرجة التي حصلت عليها ". $range ;
            }
}
?>
```

قم بوضع الملفات الثلاثة السابقة في مجلد السيرفر ثم قم بتشغيلها.

الشرح

قمت في هذا المثال بمحاولة صنع مكنكة تواصل للبيانات، بمعنى أنني أحاول أن أقوم بالاحتفاظ بالبيانات عبر الثلاث صفحات بشكل متواصل، لاحظ أنني كنت اختبر في quiz2 و quiz3 باختبار المتغيرات قبل طباعة أي شيء فقد يقوم المستخدم مثلاً بالاحتفاظ بالصفحة التي وصل إليها في المفضلة ثم يقوم بإكمال المسابقة في وقت آخر ولكني لا أريد ذلك بل أريد ان أجعل وقتها محدوداً (طبعاً هذا الكلام سيحصل إذا كانت المسابقة طويلة) لذلك فإنني عند الانتقال من صفحة إلى صفحة أقوم بإختبار إن كانت جميع هذه القيم موجودة ولاحظ أنني كنت احتفظ دوماً بقيم المتغيرات في متغيرات جديدة في حقول مخفية وكلما كان عدد المعلومات أكبر في كل مرة كان عدد الحقول المخفية أكثر، إن لهذه الطريقة أيضاً مشاكلها فقد

يفتح المستخدم كود الـhtml ويقوم بتفحص كيفية ملاحقته عبر المسابقة وقد يصنع هو الكود في وقت لاحق لكي يستطيع إكمال المسابقة بهذه الخدعة الماكرة... لذلك يفضل أن لا تقوم بذلك وتقوم بجعل المسألة السابقة أكثر تعقيداً باستخدام الـ regular expression بمحاولة تلغيم البيانات بواسطته ومن ثم فك هذا التلغيم في الصفحات التي تصل إليها البيانات.

إرسال بيانات بواسطة query strings

نستطيع إرسال بيانات بسيطة بواسطة الاستعلامات التي نقوم بإضافتها إلى اسم الصفحة في الأعلى متبوعة بـ(؟) علامة استفهام ثم اسم متغير وقيمته وإذا كان هناك أكثر من متغير يتم الربط بينهم بعلامة & وراجع فصل النماذج لمزيد من المعلومات.

قم بعمل صفحة وسمها ask.php وقم بكتابة الكود التالي: فيها:

```
<?
If (isset($ask)) {
If ($ask == login) {
Echo " تم تسجيل الدخول إلى الصفحة";
}
}
if (!isset($ask)) {
echo "لم يتم تسجيل الدخول إلى الصفحة". "<br>";
Echo "<A HREF=$PHP_SELF?ask=login> اضغط هنا ليتم تسجيل دخولك
</a><br>";
}
?>
```

قم بتجربة هذا المثال على موقع يدعم PHP على نظام تشغيل لينوكس إذا لم يعمل بشكل جيد على الويندوز

لاحظ أننا في أول الولوج إلى الصفحة لم نستخدم أي استعلامات وعند الضغط على الرابط قام الرابط بإرسال قيمة المتغير الذي يقوم الPHP باختبارها فإذا وجد أنه قد تم إرسالها (بواسطة الرابط الذي تم الضغط عليه) قام بطباعة (تم تسجيل الدخول) وإذا لم يجدها قام بطباعة (لم يتم تسجيل الدخول) بالإضافة إلى طباعة الرابط الذي يحتوي على المتغير في طياته.

الفصل الثامن
الكوكيز (cookies)

إذا ما هي الكوكيز، الكوكيز هي عبارة عن بعض المعلومات أو القطع الصغيرة من البيانات يتم الاحتفاظ بها في جهاز العميل لكي يتم الاحتفاظ بها عند الزيارات المختلفة للمستخدم (العميل)، أنت لا تقوم بالاحتفاظ فيها بقيم ضخمة لكنك تستفيد منها في أشياء أخرى مثل:

١ - جعل لكل مستخدم الألوان الخاصة التي يرى فيها صفحتك (أي أن تجعل للمستخدم مثلاً إعدادات الألوان الخاصة لرؤية موقعك).
٢ - جعل مفتاح للمستخدم لكي يستطيع به التحكم في بياناته الخاصة عند زياراته لموقعك في مرات أخرى.

الكوكيز مفيد للاستخدام في الأشياء البسيطة والغير خطيرة، لكنه الآن يستخدم بشكل سيء، مثل استخدامه مثلاً في معرفة معلومات عن المستخدم بدون علم منه، أو تخزين كميات كبيرة من البيانات فيه والتي من الأجدر أن يتم حفظها في ملف على السيرفر.
ويكون استخدامه مفيداً عندما تضمن أن جميع زوار موقعك تسمح متصفحاتهم بالكوكيز (مثل طلبة المدارس أو شبكات انترانت).
عندما يكون فقط لأشياء بسيطة لاضرر منها عند عدم السماح بالكوكيز بجهاز العميل.

بدايتك مع الكوكيز

قبل أن نبدأ علينا معرفة بعض الأساسيات عن الكوكيز

الكوكيز عبارة عن قطعة صغيرة من البيانات التي تستخدم لتخزين اسم متغير وقيمته مع معلومات حول الموقع التي أتت منه وتاريخ انتهائها.

الكوكيز عبارة عن تقنية للتخزين من جهة العميل (client-side storage) تتخزن في ملفات في جهاز العميل

يتم العبور إلى هذه الكوكيز ومسحها من المكان الذي ارسلت منه.

عندما يطلب المستعرض صفحة من السيرفر وهذه الصفحة تقوم بتخزين كوكيز فإن السيرفر يقوم بإخبار المستعرض بأنه سيقوم بوضع كوكيز للاستعمال لاحقاً.

عندما يتم طلب الصفحة في مرة أخرى يقوم المستعرض بإرسال البيانات التي تم إنشاؤها سابقاً عند طلب الصفحة.
يتم انتهاء مده الكوكيز بانتهاء وقت صلاحيتها المحدد من قبل السيرفر ويتم مسحها فورياً عند إغلاق الصفحة إذا كان وقت صلاحيتها صفراً من الثواني.

باختصار عندما يعطي السيرفر الكوكيز للمستعرض فإنه يقول لك هذا شيء أتذكرك به في وقت لاحق (قد يكون هذا الوقت من ضغط رابط آخر في الصفحة التى زرتها حتى بعد أسبوع أو أكثر).

يقوم السيرفر بإرسال الكوكيز عبر الـHTTP Headers الذي يتم إرساله قبل أي مخرج من مخرجات الـhtml
والمستعرض أيضاً يقوم بإرسال الكوكيز عبر الـHTTP Header بالإضافة إلى أن المستعرض يتعرف على من سيقوم بإرسال الكوكيز فلو كانت الكوكيز مثلاً مرسلة من قبل الموقع www.php.net فإنه لن يقوم بإرسالها إلى موقع www.phpbuilder.com.

باستطاعتك عند إنشاء الكوكيز تحديد مسار يتم إرسال الكوكيز إليه لكي يتم اقتصار عملية العبور إلى الكوكيز إلى أماكن معينة.

قبل أن نقوم بوضع كود بسيط سنقوم الآن بتعريف كيفية تخزين الكوكيز وكيفية قراءتها:

كون الـPHP لغة حديثه لعمل سكريبتات ويب فإنها تأتي بدعم كامل للكوكيز بواسطة الدالة setcookie() باستثناء أنك عند استعمالها يجب استعمالها قبل طباعة أي مخرجات html.

تأخذ الدالة setcookie() ثلاث معاملات، الثلاثة الأولى هي الأهم والأمثل استخداماً وهي بالترتيب:

- ❖ قيمة حرفية يتم تخزينها كاسم للمتغير.
- ❖ قيمة حرفية يتم تخزينها كقيمة لذلك المتغير.
- ❖ Unix timestamp الذي يقوم بالإشارة إلى تاريخ انتهاء الكوكيز.

Unix timestamp عبارة عن رقم صحيح لا يحتوي على فواصل عشرية يقوم بحساب الثواني من منتصف ليلة 01/01/1970. وإذا كنا نريد مثلاً أن نقوم بمسح الكوكيز بعد ساعة من تخزينه فإننا نقوم باستعمال الدالة time() التي تقوم بحساب الـtimestamp ثم نضيف عليه الوقت الذي نريده وفي حالتنا الساعه تساوي ٣٦٠٠ ثانية وعلى ذلك سنقوم بإضافة ناتج الدالة time على ٣٦٠٠ لكي يتم مسح الكوكيز بعد ساعة واحدة!

الثلاثة عوامل الأخرى التي يتم استخدامها أيضاً في الكوكيز ولكنها نادرة الاستخدام ولن نناقشها في موضوعنا هذا هي:

✓ المسار الذي يتم إرسال الكوكيز إليه فلو تم فتح نفس الصفحة من نفس الموقع ولكن من مسار آخر (مثلاً المسار كان pag\url\one وتم تغييره إلى

page\url\two فـإن المـستعرض لـن يقـوم بإرسـال البيانـات إلى الـصفحة لأنـه تم تحديد المسار الذي سيتم ارسال الكوكيز إليه)

- ✓ الدومين الذي سيتم إرسال البيانات إليه وهو مفيد في حالة ما إذا كان هناك أكثر من دومين تريد إرسال الكوكيز إليه.
- ✓ متغير من نوع integer يتم الإشارة إليه بـ secure يتم في حالة استخدام عمليات تشفير بالـ SSL.

العبور إلى الكوكيز بسيط جداً فالمتغير الذي يتم إرساله يتم تخزينه ضمن المتغيرات العامة (global) وعندئذ فإنه لو كان لدينا كوكيز اسمه ahmed فإن قيمته توضع مباشرة في متغير اسمه $ahmed !!

يمكننا مسح الكوكيز بأكثر من طريقة، بالطبع فإن المستخدم يستطيع مسح الكوكيز وتغيير محتوياتها بنفسه ولكن في حالة ما إذا أردنا أن نجعل السيرفر يقوم بمسحها فإننا نستخدم إحدي هاتين الطريقتين

إما أن نقوم بإخبار السيرفر بوقت قديم:

```
<?
 time()-999);، "0"،Set cookie ("ahmed"
?>
```

وإما القيام بمسح الكوكيز بكتابة اسمه فقط:

```
<?
Setcookie ("ahmed");
?>
```

مثال لتخزين وقراءة كوكيز

قم بفتح المفكرة واكتب الكود التالي:

```
<?
، $thename،If  ($thename)  setcookie ("rname"
time()+3600);
Echo '<form method="post">
<input type ="text" name="thename">
">تسجيل<input type="submit" value="
</form>';
". " ". $thename."<br><br>";قيمه المتغير الذي لديكecho "
 =". $rname ;قيمة الكوكيزecho "
?>
```

الشرح

عند تشغيل الصفحة لأول مرة

عند تشغيلك للصفحة سيتم إختبار ما إذا كان هناك متغير بالاسم $thename فإذا تم الحصول عليه فسيتم وضع قيمته في كوكيز باسم (rname) (وطبعاً لن يتم الحصول عليه في أول مرة لأننا لم نقم بإرسال أي بيانات بعد) وبعد ذلك طباعة نموذج من مربع نص واحد وزر لإرسال المعلومات.

وتتم طباعة قيمة المتغير إذا كان هناك أي متغير تم إرساله باسم $thename ويتم فحص قيمة الكوكيز $rname وطباعتها وبالطبع لا يوجد حتى الآن أي كوكيز.

المرحلة الثانية

الآن قم بكتابة أي شي في مربع النص (اكتب اسمك مثلاً) ثم قم بضغط زر الإرسال سيتم إرسال البيانات إلى نفس الصفحة ولكن هذه المرة سيتم تسجيل قيمة المتغير الذي يحمل البيانات في الكوكيز (rname) وبعد ذلك سيتم طباعة النموذج بشكل عادي وستتم طباعة قيمة المتغير $thename ولكن لن تتم طباعة قيمة المتغير $rname لأننا فقط قمنا بتسجيله ولم يتم إرساله عند طلب الصفحة (لأننا نعرف أنه يتم إرسال الكوكيز عند طلب الصفحة وهذه المرة عندما طلبنا الصفحة

لم يكن الكوكيز موجوداً بالأصل فلم يرسله السيرفر وقمنا نحن بتسجيله استعداداً للمرحلة القادمة).

المرحلة الثالثة

في هذه المرة سيكون الكوكيز موجوداً فسيتم إرساله على هيئة متغير ويتم إرساله ومن ثم طباعة النموذج وقيمة المتغير $thename وقيمة الكوكيز الذي يوجد بجهازك!

الفصل التاسع
بدايتك إلي الـ session

الـ session هي عبارة عن تقنية للترابط مع المستخدم وهي موجودة ضمن الـPHP4 ولم تكن موجودة ضمن الإصدارات التي قبله بل كان يجب أن تقوم بتركيب مكتبة لكي تستطيع استخدام هذه التقنية، يعتمد فهمنا للـsession على فهمنا للكوكيز وكيفية استعمالها ولقد تكلمنا عن الكوكيز بشكل جيد في الفصل السابق، يستخدم الـsession لعمل ميكانيكية تواصل بين المستخدم والسيرفر، فلقد قلنا أن الـhttp لا يوفر لنا ميكانيكية لعمل تواصل، فإذا طلب المستخدم صفحة من السيرفر فإن السيرفر يقوم بإعطائه ما أراد وينتهي عند ذلك فلا يعرف إن كان هو نفس المستخدم أو ليس هو... لأجل ذلك تم إنشاء تقنية الـsession لأجل عمل تقنية تواصل بين المستخدم والموقع، فباستطاعتك مثلاً أن تقوم بتحديد عدد زيارات مستخدم معين لصفحتك ليوم واحد أو لأسبوع أو لمدة معينة من الوقت.... أو يمكنك عمل متجر إلكتروني بسيط يستطيع المستخدم شراء عدة أشياء دفعة واحدة من الموقع ويكون على تواصل بينه وبين الموقع عندما يقوم بإضافة مشترى إلى سلة التسوق أو حذف مشتريات.

قبل أن أتكلم عن كيفية استخدام الـSession وإعطاء بعض الأمثلة البسيطة، سأقوم بالتكلم عن كيفية إعداد الـsession مع الـPHP.

إعدادات الـ session في الـ PHP

لكي تستطيع التعامل مع الـsession بشكل جيد يجب عليك أن تتعرف على بعض الإعدادات التي في ملف الـphp.ini

عندما تفتح الملف ستجد قسماً خاصاً فيه بالـsession هناك حوالى ١٩ إعداد ولكن لن نتطرق إليها كلها بل سنتكلم عن الأساسية والمهمة منها فقط كبداية لنا للتعرف على الـsession وكيفية عمله.

إعداد طريقة التخزين

```
session.save_handler (files | mm | user)
```

ستجد هذه العبارة مكتوبة في الملف كالتالي: بشكل افتراضي:

```
session.save_handler = files
```

وهذا الإعداد يقوم بتحديد طريقة التخزين للـsession وهناك ثلاثة حالات للتخزين:

١ - التخزين في ملفات عادية على السيرفر:

```
session.save_handler = files
```

٢ - التخزين على ذاكرة السيرفر:

```
session.save_handler = mm
```

٣ - التخزين بطريقة أخرى معرفة ومعينة من قبل المستخدم مثل التخزين في قواعد البيانات وهذا ما سوف نقوم بالتفصيل به بعد الكلام عن قواعد البيانات:

```
session.save_handler = user
```

يجب أن تأخذ في اعتبارك عدد الملفات التي سيقوم الـsession بتخزينها عند استخدامك للأعداد الأول والافتراضي خاصة عندما يكون عدد الزوار بالمئات أو الآلاف.

قد يكون استعمال الذاكرة أسرع ولكن المشكلة أنه من السهل مسح البيانات منها ببساطة.

الطريقة الثالثة قد تكون أكثر الطرق مرونة، ولكنها معقدة وصعبة جداً، وهي تعطيك مرونة لتخزين البيانات في أي وسائط مدعومة من قبل الـPHP مثل قواعد بيانات mysql و oracle.

الذي افترضه الآن أنك قمت بوضع قيمة هذه الخاصية إلى files .

إعداد مكان التخزين

```
session.save_path (path/to/directory)
```

هذه الخاصية مفيدة إذا كنت قد ضبط الإعداد السابق إلى files

تقوم هـذه الخاصـية بتحديـد مكـان التخـزين علـى الـسيرفر ومـن الأفـضل أن تقـوم بتحديد مكان التخزين بعيداً عن مجلد السيرفر لكي تمنع تصفح هذه الملفات.

الإنشاء التلقائي للـ session

```
session.auto_start (0 | 1)
```

هذا الإعداد يقوم بتحديـد إذا مـا كـان الـsession سيتم إنشاؤه تلقائيـاً عنـد كـل زيـارة للموقـع أو لأي صـفحة مـن صـفحاته بـدون إدراج كـود الــsession في كـل صفحة... وعلى ذلك فإنك تقوم بوضع القيمة إلى (١) إذا أردت ذلك.

وعلى افتراض أنك لا تحتاج إلى أن تجعل الـPHP يقوم بعمل session لكل صفحة تلقائياً ومن غير طلب فستقوم بوضع قيمة هذا الإعداد إلى (٠).

الـ SID

عندما يقوم الزائر بزيارة صفحتك فإن الـsession يستطيع تتبع هـذا الزائـر وعـدد المـرات الـتي قـام فيهـا الزائـر بالـدخول لليـوم الواحـد، يقـوم الــPHP بعمـل SID (session identefier) أو رقم معرف تلقائي بشكل افتراضي عندما تقوم بطلب إنشاء session بالزائر، وكل رقم معرف يختلف عن الآخر تماماً، إن رقـم المعـرف الذي ينشئه الـ PHP شبيه للشكل التالي:

```
fc94ad8b1ee49ef79c713ee98ac1fcc4
```

هناك طريقتان يستطيع بهما الـ PHP متابعة الـ SID للمستخدم:

١ - عن طريق المتابعة والتخزين بتسلسل في الكوكيز.

٢ - عن طريق اتباع رقم المعرف بعنوان الصفحة في الإنترنت.

سنأخذ أمثلة عن كلا الطريقتين:

١ - استخدام الكوكيز

بالطبع هذه هي أكثر الطرق شيوعاً للحصول على ترابط بين المستخدم والموقع وهي الأسهل، ولكن يجب أن تضع في اعتبارك أن المستخدم قد يكون قد ألغى أو منع ميزة الكوكيز في المتصفح أو قد يكون متصفحه لا يدعم الكوكيز.
خذ في اعتبارك أن بعض المتصفحات لا تسمح بأن يزيد حجم الكوكيز عن ٥ كيلوبايت.

هناك بعض الإعدادات البسيطة في ملف php.ini التي يجب معرفة معلومات عنها قبل البدء باستخدام الـsession مع الكوكيز:

```
session.use_cookies (0 | 1)
```

هذه الخاصية تحدد ما إذا كان يمكنك استخدام الكوكيز مع الـ session أو لا وعند وضع القيمة (٠) فهذا يمنعك من استخدام الكوكيز مع الـ (session)
وأما إذا كانت قيمته (١) فهذا يسمح باستخدام الكوكيز مع الـ session

```
session.name (Default: PHPSESSID)
```

هذا الإعداد يقوم بتحديد اسم الكوكيز الذي سيحتفظ برقم المعرف (SID) والإعداد الافتراضي هوPHPSESSID ولن أقوم بتغيير هذا الإعداد لكي تستطيع فهم المثال الذي سأطرحه بعد قليل.

```
session.cookie_lifetime (Default: 0)
```

يقوم هذا الإعداد بتحديد المدة التي سيبقى فيها الكوكيز الذي يحتفظ بقيمة الـ(SID) والإعداد الافتراضي هو صفر، أي أنه سيتم مسح الكوكيز تلقائياً بعد إغلاق المستخدم لنافذة المتصفح مباشرة.

```
session.cookie_path (Default: /)
```

يقوم هذا الإعداد بتحديد مسار دومين يتم تخزين الكوكيز له.. لا تقم بتغيير قيمته ودعه كما هو.

```
session.cookie_domain (Default: null)
```

يقوم هذا الإعداد بتعريف اسم دومين يتم تخزين الكوكيز لصالحه.. والقيمة الافتراضية هي null، لا تقم بتغييرها

ضع في اعتبارك أنه إذا كانت قيمه الأعداد (session.use_cookies) تساوي واحد فإنه لا داعي لاستدعاء الدالة set_cookie() لإعداد الكوكيز بل سيتم إعدادها تلقائياً بواسطة الـPHP

٢ - الإضافة أوالكتابة إلى عنوان الصفحة

إن إضافة عنوان الـSID إلى عنوان الصفحة يعتبر من الأشياء البشعة جداً رغم أن طريقته سهلة ومفيدة في حالة ما إذا كان الكوكيز غير مدعوم في المتصفح بشكل جيد.

مثال:

```
<a href="configure.php?<?=SID?>">Go to the configuration page</a>
```

بهذه الطريقة نقوم بإضافة المتغير المرجعي SID الذي سيقوم بإعطاء رقم معرف للمستخدم.

متابعة الـ session

لقد أخذنا حتى الآن معلومات تجعلنا ندخل عالم البرامج المسيرة بالـ session بدون خوف، سأبدأ الآن بطرح بعض الأمثلة البسيطة التي تثبت لديك بعض المفاهيم الأساسية في الـ session... سأشرح في هذا المثال كيفية إنشاء الـ SID وتخزينه لاستعماله لاحقاً، وخلاصة السيناريو للصفحة أننا نريد من المستخدم أن يفهم أنه يستطيع تخصيص لون الخلفية الذي يريد أن يشاهد به صفحات موقعنا... سأقوم

بتخزين قيمة مبدئية في المتغير الذي يقوم بتحديد لون الصفحة ، أنا أفترض طبعاً أن المتصفح يدعم الكوكيز:

سكريبت يقوم بانشاء وتسجيل متغير **session**

```
<?
session_start();
session_register("zx");
session_register("co");
$zx=10;
$co++;
echo 'مرحباً بك في موقعنا أيها الزائر الكريم'."<br>";

echo "عدد زياراتك لهذه الصفحة =". $co ;
echo "<br>";
echo '<a href="php2.php">الصفحة الثانية</a>';
?>
```

أقصد بالجلسة هي الـ(session) وإن كانت الترجمة غير صحيحة ولكن فقط نأخذه كمصطلح.

متغير الجلسة هو الـ(session-variable) أو متغير الـsession أو سمه ما شئت.

الشرح

يقوم هذا السكريبت في البداية بإنشاء متغير اسمه (ZX) ومتغير اسمه (CO) وقمنا بإعطاء القيمة (١٠) للمتغير (ZX) وقمنا بزيادة القيمة الموجودة (وهي الصفر) في (CO) بواحد وكتبنا مرحباً بك أيها الزائر الكريم في موقعنا ، ثم قلنا له إن عدد زياراتك لهذه الصفحة هي قيمة المتغير(CO) ثم أعطيناه رابط للصفحة الثانية.

في الواقع إن هذه المتغيرات وقيمها يتم الاحتفاظ بها في كوكيز له اسم خاص قمنا بتحديده سابقاً من ملف PHP.ini ، وهذا الكوكيز يحتفظ بقيمة الـSID للـsession.

نحن لا نقوم بإخبار الـPHP أين سيحتفظ بقيمة المتغيرات لأننا بدأنا بكلمة الـ:

```
session_start();
```

وعلى هذا فإن الـPHP سيفهم أنه سيقوم بتخزين القيمة في الكوكيز الخاص بالـsession.

قمنا بجعل المتغير co كعداد بسيط لعدد المرات التي سوف نقوم بها بزيارة الصفحة فعند عمل تحديث للصفحة سيتم زيادة العداد بمقدار واحد

```
$c++;
```

وطبعاً قبل زيادة العداد بقيمة واحد فإنه يتم حساب القيمة السابقة للمتغير عند إنشائه تلقائياً... ومن ثم تتم الزيادة وبعد ذلك طباعة القيمة.

كتابة رقم الـ SID

اكتب الآن الكود التالي واحفظه باسم php2.php

```
<?
session_start();
echo $PHPSESSID."<br>";
echo $zx;
?>
```

في هذه الصفحة نقوم بطباعة قيمة الـSID وذلك بطباعة قيمة المتغير $PHPSESSID (الذي هو اسم الكوكيز الخاص بالـsession).

بعد ذلك قمنا في النهاية بطباعة قيمة المتغير $zx لكي ألفت نظرك بأن الكوكيز ما زال يحتفظ بها ولم يفقدها لأننا قد حددنا الإعداد في ملف php.ini الخاص بوقت الكوكيز الـ 3600 أي لمدة ساعة ثم بعد تلك الساعة سيتم مسح الكوكيز ولن يمكنك من استرجاع قيمة أي متغير:

```
session.cookie_lifetime = 3600
```

واضف إلى معلوماتك أنه لا يمكنك قراءة القيم للكوكيز الخاص بالـsession إلا عن طريق إضافة الأمر

```
session_start();
```

يجب أن تبدأ بهذا الأمر دائماً إذا أردت قراءة قيم المتغيرات التي يحتفظ بها الكوكيز الخاص بالـsession.

مسح متغ   من الـ session

كل ما عليك فعله هو استخدام هذه الدالة:

```
session_unregister(variable name);
```

تقوم بوضع اسم المتغير في مكان الـ(variable name)

مثال:

```
session_unregister(“brn”);
```

سيقوم هذا الأمر بمسح المتغير (brn) من الـكوكيز الخاص بالـ(session)

قراءة قيم المتغيرات في الكوكيز الخاصة بالـ session

كل ما عليك فعله هو استخدام الدالة:

```
session_encode();
```

مثال:

```
<?
session_start();
session_register("bgcolor");
session_register("name");
session_register("email");
$bgcolor = "#8080ff";
$name = "alfareees almolthem";
$email = "php@php.com";
$e = session_encode();
print "The encoded string is: $e";
?>
```

بهذا السكريبت نكون قد أنهينا فصلنا عن مقدمة بسيطة للـsession. هذه مجرد مقدمة ولكي نستطيع أن نتعمق بالـsession فيجب علينا أن نتعلم شيئاً عن قواعد البيانات.

الفصل العاشر
اختصارات هامة

- عبارة الـ if.
- عبارة الـ else.
- عبارة الـ elseif.
- عبارة الـ switch.
- حلقة التكرار while.
- حلقة التكرار for.
- حلقة التكرار do while.

- عبارة الـ if:

استخدام الـ if في كتابة السكربات شيء أساسي، وكما في لغات البرمجة الأخرى فإن الـ PHP تتبع نفس الأسلوب في كتابة الـ if، فيمكن تحديد شرط معين مقترن بالـ if وبالتالي إذا كان الشرط صحيحاً (true) فسيتم تنفيذ الأسطر المحددة، وبتفصيل أكثر يجب وضع الشرط بين قوسين ()، ووضع الأسطر المطلوب تنفيذها بين العلامات { }، مع ملاحظة أنه يمكن التخلي عن العلامات { } في حال وجود سطر واحد فقط.

فلنفترض وجود نموذج بريدي (Mail Fourm)، يحتوي على الاسم والبريد والرسالة، ونرغب في معرفة ما إذا كان المرسل قد ملأ جميع الحقول وبالتالي إرسال الرسالة، أو أنه لم يفعل ذلك وبالتالي عرض رسالة (فضلاً قم بتعبئة البيانات كاملة)، لعمل ذلك نحتاج لمعرفة أسماء المتغيرات في النموذج، ولذلك فلنفترض أن المتغيرات كالتالي:

(الاسـم $name)، (البريـد $email)، (الرسـالة $later)، ولعمـل الـشرط الأول (إذا كان الاسم لم يُدخل فلن يتم إرسال الرسـالة):

```
<?
 if ($name == "")
 "; فضلاً قم بتعبئة البيانات كاملة echo "
 ?>
```

والمعنى أنه إذا كان المتغير $name لا يحتوي على أي قيمة (أي فراغ) فسيتم تنفيذ السطر التالي وطباعة الجملة، مع ملاحظة أن المطلوب تنفيذه هو سطر واحد فقط ولـذلك لم نـستخدم { }، بـل في حالـة وجـود أكثـر مـن سـطر يجـب اسـتخدامها كالتالي:

```
<?
if ($name == "")  {
<br> "; فضلاً قم بتعبئة البيانات كاملة echo "
 "; لم تقم بإدخال الاسم echo "
}
?>
```

- عبارة الـ else:

هذه العبارة تتيح إمكانية وجود إجراء ثاني لعدم تحقق الشرط، ففـي مثالنـا الـسابق كان الإجراء طباعة الجملة إذا تحقق الـشرط، ولكـن في حالـة عـدم تحقـق الـشرط فلن يكون هناك إجراء لتنفيذه، بل إن الإجـراء سـيتم تنفيـذه إذا تحقـق الـشرط ومـن ثم سـيتم إكمـال بقيـة الأسـطر، وفي حالـة مثـل هـذه الحالـة يـتم اسـتخدام الــ else لوضع إجراء آخر في حالة عدم تحقق الشرط، وبالمثال يتضح المقال:

```
<?
if ($name == "")  {
"; فضلاً قم بتعبئة البيانات كاملة echo "
}
else
{
"; تم إرسال الرسالة، شكراً لك echo "
```

```
}
?>
```

في هـذا المثـال سـتتم طباعـة الجملـة (فـضلاً قـم بتعبئـة البيـانات كاملـة) إذا تحقـق الشرط أن المتغير $name لا يحتوي على أي قيمة، وستتم طباعة الجملة (تم إرسـال الرسـالة، شكراً لك) في حالة عدم تحقق الشرط، أي في حالة وجود قيمة في المتغير $name، مـع ملاحظـة أن هـذا المثـال يحتـوي علـى شـرطين ولـيس شـرط واحـد، فالظاهر هـو شـرط واحـد ("" == $name) ولكـن العبـارة else تعتبر شـرطاً بحـد ذاتها ولو لم يكن هذا الشرط مكتوباً، وكما هـو واضح فمعنـى هـذا الـشرط هـو (إذا كان غير ذلك) فقم بطباعة الجملة.

يمكن أن يكون الشرح غير واضح تماماً، ولكـن أهميـة فهم الطريقـة ستتضح في الأسطر القليلة القادمة.

- عبارة الـ elseif:

في العبـارة الـسابقة ذكرنـا أنـه يوجـد شـرطين وإجـرائين، أحـد هـذين الـشرطين غـير مكتوب بل هو مفهوم من إدراج العبـارة else، وفي حـالات كـثيرة لا يكفينـا مجـرد شـرطين واجـرائين لاتمـام بعـض الـسكربات المعقـدة، فلـذلك يمكـن أن نـستخدم العبـارة elseif مع الـ if لعمـل مثـل هـذه الـسكريبتات، فلـو افترضـنا أن لـدينا عـداد لزوار الموقع ونريد إظاهر العداد بحيث تتم قراءتـه بشكل جيد، أي بمعنـى آخـر إذا كان عدد الزوار (١) فستتم طباعة الجملة (عدد الزوار: زائر واحد فقـط) وإذا كـان (٢) فستتم طباعـة الجملـة (عـدد الـزوار: زائـرين)... وقـس علـى ذلـك، فعنـدما يكـون عـدد الـزوار (١) سـيتم عـرض الجملـة الأولى فقـط وعنـدما يكـون عـدد الـزوار (٢) فسيتم عرض الجملة الثانية فقط، وهكذا لبقية الشروط.

بافتراض أن المتغير ($counter) هو عداد الـزوار، فالمثـال التـالي يبين مـا تم شـرحه سـابقاً:

```
<?
if ($counter == 1)  {
echo "عدد الزوار: زائر واحد فقط ";
}
elseif ($counter == 2)  {
echo "عدد الزوار: زائرين ";
}
elseif ($counter >= 3 && $counter <= 10)  {
echo "عدد الزوار: $counter زوار ";
}
else {
echo "عدد الزوار: $counter زائر ";
}
?>
```

كما هو واضح في المثال السابق سيتم ما يلي:

الشرط: العداد يساوي ١

الإجراء: طباعة (عدد الزوار: زائر واحد فقط)

الشرط: العداد يساوي ٢

الإجراء: طباعة (عدد الزوار: زائرين)

الشرط: العداد أكبر أو يساوي ٣ و اصغر أو يساوي ١٠

الإجراء: طباعة (عدد الزوار: (العداد) زوار)

الشرط: العداد لا يحقق أي من الشروط

الإجراء: طباعة (عدد الزوار: (العداد) زائر)

ملاحظة بسيطة فقط، وهي على العلامة && التي تعني (و)، وهي من علامات الجمع بين جملتين، فيجب أن تكون الجملتين صحيحتين لتحقق الشرط.

- عبارة الـ switch:

هذه العبارة قريبة جداً من العبارة if، ولكن يمكن استخدام أكثر من شرطين بأسلوب آخر، غير أنه يجب إسناد قيمة معينة للـ case وهي هنا بمثابة الشرط، لكي يتم تنفيذ الإجراء المحدد لذلك الشرط أو الـ case، وفي النهاية الأمر يعود إلى المصمم وأيهما يفضل، وكما في المثال السابق يمكن كتابة مثال بالـ switch بنفس الطريقة، والمشكلة الوحيدة هي كما قلنا أنه يجب إسناد قيمة معينة لكل case وبالتالي فإن الشرط الثالث من المثال السابق يجب تفريقه لكل قيمة من (٣ إلى ١٠)، وهذه العملية مجهدة لانه يجب كتابة سطر لكل قيمة كما يلي:

القيمة: ٣

الـ case: 3

الإجراء: طباعة (عدد الزوار: (العداد) زوار)

القيمة: ٤

الـ case: 4

الإجراء: طباعة (عدد الزوار: (العداد) زوار)

القيمة: ٥

الـ case: 5

الإجراء: طباعة (عدد الزوار: (العداد) زوار)

.... الخ...

وفي المثال التالي سأتغاضى عن الشرط الثالث بكامله، وأذكر بقية الشروط والحالات لمجرد فهم طريقة عمل هذه العبارة:

```
<?
switch ($counter)
{
case "1";
echo " عدد الزوار: زائر واحد فقط ";
```

```
break;
case "2";
echo "عدد الزوار: زائرين ";
break;
default;
echo "عدد الزوار: $counter زائر ";
break;
}
?>
```

استخدمنا في هذه المثال بعض الجمل وتعني ما يلي:

Switch وتكتب في البداية مع إدراج اسم المتغير الذي سيتم عمل الشروط عليه.

Case أي في حالة (....) ويكتب بجانبها الشرط.

Break وتعني إيقاف العملية والخروج من الشرط بعد تنفيذ أحد الإجراءات.

Defaukt وهي تقابل العبارة else أي بمعنى أنها لأي حالة لم يتم ذكرها في الشروط.

- حلقة التكرار while:

وهي أبسط حلقات التكرار على الإطلاق، بحيث تأخذ شرط واحد فقط وتبني على تنفيذ ما بين علامات الشروط { }، والفرق الوحيد بينها وبين الـ if هو أنها ستقوم بتنفيذ الإجراءات طالما كان الشرط صحيحاً، وهذا يعني احتمال تنفيذ الإجراء أكثر من مرة، وهذه الدالة مفيدة في إدراج الحقول من الجدوال وغيرها من الاستخدامات، بحيث لو افترضنا وجود جدول معين في قاعدة بيانات ونريد إدراجه في صفحة PHP، فسيكون من أهم خيارات استخدام هذه الدالة، وباذن الله سيتم التطرق لقواعد البيانات في الفصول القادمة، وفي الوقت الحالي سأذكر مثال بسيط على هذه الدالة لفهم طريقة استخدامها:

```
<?
$total = 10;
while ($total <= 50)
{
```

```
echo "العدد أقل من ٥٠ <br>";
$total +=10;
}
?>
```

كير بسيط يمكن معرفة أن الجملة (العدد أقل من ٥٠) ستتم طباعتها ٥ مرات، لأن حلقة التكرار while قامت بتنفيذ الاجراء طالما أن الشرط صحيح، وفي المرة الأولى كان المتغير ($total) يساوي (١٠) والشرط صحيح لأن الـ ($total) فعلاً أصغر أو يساوي الـ (٥٠)، فتم تنفيذ ما بين علامات الشرط، ومن ذلك زيادة متغير المجموع ($total) بقيمة (١٠) ومن ثم الرجوع والمقارنة من جديد، وفي هذه الحالة صار المتغير ($total) يساوي (٢٠) وأيضاً الشرط صحيح وبالتالي الدخول مرة أخرى وتنفيذ الأجراء.... وهكذا حتى يتم الوصول إلى أن قيمة الـ ($total) يساوي (٥٠) وبالتالي الشرط صحيح، ومن ثم تصبح قيمة الـ ($total) تساوي (٦٠) وفي هذه الحالة يتم ايقاف تنفيذ الإجراءات لأن الشرط غير صحيح.

- حلقة التكرار for:

يوجد طريقة أسهل للتعامل مع المثال السابق، فاستخدام حلقة التكرار while كانت القيمة الابتدائية للمتغير ($total) في سطر، والشرط في سطر والزيادة على المتغير في سطر أخر، وبالتالي زيارة في عدد الأسطر عن ما يمكن استخدامه مع حلقة التكرار for، فالمثال التالي يبين طريقة أخرى لاستخدام مثال الـ while بطريقة أسهل:

```
<?
for ($total = 10;  $total <=50; $total +=10)
{
echo "العدد أقل من ٥٠ <br>";
}
?>
```

وللتوضيح فإن تركيب الـ for هو على الشكل التالي:

```
for(القيمة الافتراضية؛ الشرط؛ مقدار الزيادة ;)
{
الإجراء المطلوب تنفيذه
}
```

- حلقة التكرار do while:

وهي نسخة أخرى من الـ while والفرق الوحيد بينهما أن التأكد من الشرط وصحته من عدمها يتم بعد تنفيذ الإجراء وليس قبله كما في الـ while وكمثال عليها:

```
<?
$total = 10;
do
{
echo "العدد أقل من ٥٠ <br>";
$total +=10;
}
while ($total <= 50);
?>
```

الفصل الحادي عشر
قواعد البيانات والـ Mysql

في البداية سنتعرف على مصطلح الـ RDBMS، ونعني بذلك قواعد البيانات العلائقية، والتي من خصائصها سهولة الوصول إلى البيانات المخزنة فيها، وسرعة إتمام عمليات الاستعلام المختلفة، وبالاضافة إلى المميزات الأخرى فإن هذا النوع يعتبر الأكثر استخداماً في جميع التطبيقات سواء المستخدمة في الإنترنت أو ذات الطابع البرمجي الخاص، وبطبيعة الحال فإن الـ Mysql من هذا النوع.

ومن المهم معرفة بعض الأساسيات في الـ RDBMS، والتي من شأنها تسهيل عملية فهمك التام لطريقة عملها والتعامل معها..

١- الجداول Tables:

تعتبر أكبر جزء في قاعد البيانات، وهي عبارة عن أعمدة وصفوف تحتوي على قيم معينة.

٢- الأعمدة Columns:

لكل عمود في الجدول اسم خاص يختلف عن أسماء الأعمدة الأخرى في نفس الجدول، ويجب أن يكون لكل عمود نوع خاص به يصف نوع البيانات التي ستخزن فيه، وكم يظهر في الصورة، فإن عمود الرقم من النوع الرقمي Integer، أما الحقلين الآخرين فهي نصوص Text.

٣- الصفوف Rows:

كل صف من صفوف الجدول يحتوي على قيم مختلفة ويمثل معلومات متكاملة عن قطاع معين، وفي مثالنا يمثل معلومات متكاملة عن شخص معين.

٤- القيم **Values**:

وهي ما تحتوي عليه تقاطعات الصفوف بالأعمدة.

٥- المفاتيح **Keys**:

وتعتبر من أساليب تسهيل الوصول إلى المعلومات في قواعد البيانات، وفي مثالنا السابق نرى أن العمود Id يحتوي على أرقام متسلسلة لا تتكرر نهائياً بل إنها تتكون بشكل تلقائي عند إدراج أي صف جديد للجدول، وبالتالي فإنها تعتبر المفتاح المناسب لكل صف من صفوف الجدول لضمان عدم الالتباس في اختيار الصفوف.

فلو افترضنا أن لدينا جدولين في قاعدة بيانات، يحتوي الجدول الأول على معلومات عن الدروس مفصلة على عدة حقول لتلك الدروس، على سبيل المثال:
الرقم (id)، الدرس (lesson)، رقم الكاتب (Key_author)..
ويحتوي الجدول الثاني على بيانات الأعضاء كما يلي:
الرقم (Key_author)، الاسم (name)..

والمطلوب هو طريقة لربط الجدولين، بحيث أن رقم الكاتب في جدول الدروس (Key_author) يدل على اسم الكاتب في جدول الأعضاء (name).

بالتدقيق في المثال يتضح أن الحقلين (أو العمودين) Key_author في كلا الجدولين هو مفتاح الربط بينهما، ولذلك يمكن الوصول إلى اسم الكاتب اعتماداً على رقمه من جدول الدروس، وبالتالي الربط بين الجدولين.

لن أتحدث طويلاً عن مقدمات قواعد البيانات Mysql، ولكن بهذه المقدمة البسيطة يمكن على الأقل تصور بعض الأساسيات حول قواعد البيانات عموماً والـ Mysql خصوصاً، ومن وجهة نظري فالاهم هو كيفية التعامل مع قواعد البيانات

بما يخدم احتياجاتنا مع الـ PHP، ولذلك سأتطرق في هذا الفصل إلى نقطة هامة جداً وهي إدارة قواعد البيانات، وأعني بذلك عملية إنشاء قواعد البيانات والجداول والتحكم في الحقول والبيانات وغيرها، لتكون الأساس للتعامل مع قواعد البيانات لاحقاً عن طريق الـ PHP، ولعمل ذلك يوجد عدة طرق من أهمها الطريقة التقليدية المباشرة بالاعتماد على نظام الدوس في ذلك وبدون استخدام أي برامج أخرى للإدارة.

الاتصال بالـ Mysql، والتعامل معها:

كما قلنا أن الطريقة التقليدية هي الاتصال بقواعد البيانات عن طريق سيرفر الـ Mysql وبدون استخدام أي مكونات أخرى، ولعمل ذلك نحتاج أن نعرف مسار سيرفر الـ Mysql على الجهاز المستخدم بعد عملية التثبيت، كما قمنا بذلك في فصل المقدمة، وعادة يكون المسار كالتالي (C:\mysql\bin)، وبذلك يمكن تشغيل البرنامج mysql.exe من داخل الـ Dos.

عموماً طريقة الاتصال بقاعدة البيانات هي كالتالي:

```
mysql –h HostName –u UserName –p
```

مع استبدال الـ HostName باسم السيرفر لديك، سواء كان السيرفر على نفس الجهاز وفي هذه الحالة تكتب localhost، أو أن السيرفر الذي تود الاتصال به ليس على نفس الجهاز وبذلك تكتب المسار الكامل لاسم السيرفر (HostName)، ومع استبدال الـ UserName باسم المستخدم الخاص بالـ Mysql لديك، بعد ذلك سيتم طلب كلمة المرور الخاصة بقاعدة البيانات بعد الضغط على Enter، قم بإدخالها وسيتم فتح الاتصال بالـ Mysql، كما يمكن كتابة mysql فقط ليتم فتح الاتصال بقاعدة البيانات فقط إذا كنت تعمل على نفس الجهاز وليس جهاز أخر.

سيظهر المؤشر الخاص بأوامر الـ Mysql كالتالي:

```
mysql>
```

وبهذا نكون وصلنا إلى المكان المطلوب لكتابة أوامر الـ Mysql والتحكم بها.

الأمر الأول الذي سنقوم بكتابته يقوم باستعراض قواعد البيانات الموجودة على السيرفر والأمر هو:

```
show databases;
```

بعد كتابة هذا الأمر (بعد مؤشر الـ mysql>)، سيتم استعراض قواعد البيانات في السيرفر الذي قمنا بالاتصال به، وفي حالة عدم وجود أي قاعدة بيانات قمت بإعدادها من قبل، فان من الطبيعي أن تجد قاعدتي بيانات موجودة بشكل تلقائي عند تثبيت السيرفر Mysql، وتلك القاعدتان هما test – mysql.

ولمحاولة فهم الموضوع بشكل أكبر، سنقوم بالتطرق إلى مثال يبين كيفية إنشاء قاعدة بيانات، وكيفية الدخول لها والتعامل معها وإنشاء الجداول، ومن ثم حذفها..

بعد استعراض قواعد البيانات بالأمر السابق، سنقوم بانشاء قاعدة بيانات باسم PHP، ولعمل ذلك قم بكتابة الأمر التالي:

```
create database PHP;
```

لو قمنا بكتابة الأمر السابق (show database) سنرى أن قواعد البيانات أصبحت ٣ بإضافة القاعدة PHP إلى القاعدتين mysql – test، ولاستخدام أي منها نقوم بكتابة الأمر التالي في مثالنا مع القاعدة PHP:

```
use PHP;
```

وهذه يعني الدخول في قاعدة البيانات PHP واستخدام المؤشر (mysql>) لكتابة الأوامر المتعلقة بالتعامل مع قاعدة بيانات بعينها.

أول هذه الأوامر هو أمر إنشاء جدول في قاعدة البيانات، وهذه الأمر يحتاج إلى تفصيل دقيق لبعض الخصائص مثل أسماء الحقول وأنواع البيانات فيها، وبعض

الأشياء الأخرى، عموماً قم بكتابة الأمر التالي وسأقوم بشرح كافة التفاصيل بعد المثال:

```
create table users(
id Int not null auto_increment Primary Key,
name text not null,
counter int
);
```

شرح المثال:

- قمنا بكتابة (create table users) وهذا يعني إنشاء جدول باسم users.
- القوس) يعني بداية تسمية حقول الجدول وخصائص تلك الحقول.
- السطر الأول من أسماء الحقول هو (id) والرمز (int) يعني وصف نوع البيانات التي ستخزن في الحقل (id)، وهي في هذه الحالة تعني نوع البيانات الرقمية، أما الرمز (not null) فيعني عدم إمكانية أن يكون هذا الحقل فارغاً، بل يجب أن يحتوي على قيمة، والـ (auto_increment) يجعل الحقل يحتوي على قيم متسلسلة يستحيل تكرارها، وسيبدأ من الرقم ١ ويبدأ بالزيادة بمقدار واحد في كل مرة يتم إدخال صف جديد إلى هذا الجدول، وفي النهاية الرمز (Primary Key) يعني أن الحقل هو المفتاح الرئيسي لهذا الجدول أو بمعنى إنه سيتم التفريق بين صفوف الجدول اعتماداً على هذا الحقل ولهذا وضعنا (auto_increment) لضمان عدم اختلاط البيانات.
- السطر الثاني يحتوي على اسم الحقل (name) ونوع البيانات (text) أي نصي، ونفس الرمز السابق الذي ذكرناه وهو (not null).
- السطر الثالث يحتوي على اسم الحقل (counter) ونوع البيانات (int)، ولاحظ أننا لم نذكر (not null) وبالتالي يمكن أن يكون هذا الحقل فارغاً لا يحتوي على أي قيمة، ولن يكون هناك أي تعارض أو مشكلة بعكس الحقلين السابقتين.
- في السطر قبل الأخير، أي قبل علامة الإغلاق (، سيكون بدون فاصلة.
- السطر الأخير يحتوي على أقفال عملية إنشاء الجدول بالعلامة)؛.

عموماً هذا المثال يعطي نبذة بسيطة عن كيفية إجراء مثل هذه الأوامر، وسنتطرق إلى بقية الأوامر في الأسطر القليلة القادمة.

يمكنك استعراض الجداول الموجودة في قاعدة بيانات عن طريق الأمر:

```
show tables;
```

ولو قمت بتطبيق ذلك على المثال السابق فسترى أن الجدول users موجود في قاعدة البيانات PHP التي قمنا بإنشائها.

يمكن كذلك استعراض خصائص الجدول السابق users الذي قمنا بإنشائه في المثال السابق، عن طريق الأمر التالي:

```
describe users;
```

سترى أن حقول الجدول وخصائص كل جدول ظهرت لك بشكل واضح.

- التعامل مع بيانات الجداول:

بقي أن نذكر الطرق التي يمكن من خلالها إدخال البيانات إلى الجدول users، بل وكيفية التعامل مع تلك البيانات بالتعديل والحذف وغير ذلك، وكما قلنا سابقاً إن هذه الأساسيات مفيدة جداً في البرمجة بلغة الـ PHP، بل إن فهم هذه الطرق هو المفتاح الأساسي للتعامل مع قواعد البيانات عن طريق الــ PHP،

عموماً إن أول تلك الأوامر هو إضافة صف جديد إلى الجدول، وهذا ما يبينه المثال التالي:

```
insert into users set
name = “Ahmad”;
counter = 3
;
```

مع ملاحظة أن users هو اسم الجدول، name اسم الحقل (العمود) الأول، counter اسم الحقل (العمود) الثاني، كما تلاحظ أن الحقل id لم نتطرق له،

لأننا في إعدادنا للجدول ذكرنا أن الحقل (id auto_increment) أي ستضاف إليه القيم بشكل تلقائي وبشكل منظم، كما قلنا في كل مرة يزيد العداد بقيمة ١، و بطبعية الحال يمكنك القياس على هذا المثال باستبدال ما يجب استبداله من اسم الجدول (users) واسماء الحقول (name – counter) وكذلك البيانات بما يناسب الذي تريد القيام به.

هذا بالنسبة لإضافة بيانات جديدة إلى جدول معين، أما بالنسبة لاستعراض البيانات في الجدول فكما يلي:

```
select * from users;
```

ومعني select (اختر)، ولذلك ستجد أن جميع البيانات التي في الجدول users قد تم سردها، واذا كنت ملتزماً بالمثال السابق حرفياً فستجد أن البيانات التي أضفناها في المثال السابق ظهرت على شكل صف من صفوف الجدول، وبالتالي كلما أضفت صفاً جديداً إلى الجدول وقمت باستعراض البيانات تجد أن بياناتك قد تم تخزينها، وينطبق الكلام السابق حول الاستبدال هنا أيضاً، فيمكن استبدال اسم الجدول users بأي اسم لجدول في قاعدة البيانات المستخدمة، وللتأكد من أسماء الجداول قم باستخدام الطريقة السابق ذكرها وهي (show tables).

النقطة الأخيرة التي سأتطرق لها هي ما يجب معرفته حول الأمر select وهو كثرة استخدامه في التعامل عن طريق الـ PHP، وبالتالي يجب عليك فهم طريقة كتابته بشكل كامل، بالاضافة إلى خيارات الاختيار إن صح التعبير، وهي ما يتم كتابته بعد الجملة السابقة من خيارات تحدد طريقة اختيار البيانات من شروط وترتيب وحدود وهذا ما ساذكره في الاسطر القليلة القادمة.

فلفترض أن الجدول السابق يحتوي على أكثر من صف من البيانات بالشكل التالي:

أما البيانات التي نود جلبها فهي كما يلي لكل نقطة على حدة:

١- بيانات الأعضاء الذين ليس لهم أي موضوع.

٢- بيانات الأعضاء الذين لهم مواضيع أكثر من ٥ مرتبين من الأكثر إلى الأقل.

٣- بيانات العضو Ahmed.

٤- بيانات جميع الأعضاء مرتبين حسب الاسم.

٥- بيانات العضو الأكثر مواضيعاً.

سنأخذ كل حالة على حدة:

الحالة الأولى:

يمكن التعامل معها كما يلي:

```
select * from users where counter=0;
```

الزيادة التي قمنا بوضعها هي (where counter=0) أي بحيث أن الحقل (counter) يساوي صفر، وبالتالي سيتم إهمال أي صف من البيانات التي لا يحتوي الحقل (counter) فيها على القيمة صفر، وسيتم جلب البيانات التي يحتوي هذا الحقل فيها على صفر.

الحالة الثانية:

```
select * from users where counter >= 5 order by counter;
```

في هذا المثال أضفنا الشرط (where counter >= ٥) وهو واضح كما في المثال السابق ولكن تم تغيير الشرط لا أقل ولا أكثر، أما الإضافة الأخرى فهي طريقة الترتيب وهي (order by counter) وتعني (قم بترتيب البيانات المختارة بحسب الحقل counter)، وهناك طريقة أخرى للتحكم في الترتيب اما تصاعدي أو تنازلي وذلك بإضافة كلمة asc ليكون الترتيب تنازلياً كما هو الحال في المثال السابق، فسواء ذكرت ذلك أو سيتم اعتبارها تنازلياً بشكل تلقائي، أما الأهم فهو

طريقة الترتيب التصاعدي من الأقل إلى الأكبر ويتم ذلك عن طريق كتابة الكلمة desc بعد الترتيب مباشرة لتصبح كما يلي:

```
select * from users where counter >= 5 order by counter desc;
```

الحالة الثالثة:

```
select * from users where name = “Ahmed”;
```

لاحظ أن الفرق الوحيد هنا هو استخدام علامات التنصيص، لأن نوع البيانات نصية.

الحالة الرابعة:

```
select * from users order by name;
```

وقد أوردت هذا المثال لبيان أنه يمكن استخدام أحد الخيارات لجلب البيانات وترك باقي الخيارات، فيمكن كما في المثال استخدام خيار الترتيب (order) وعدم استخدام الخيارات الباقية (where – limit)، أما الخيار where فقد تطرقنا له سابقاً وتعرفنا على فائدته، والخيار الأخر limit هو ما سيتم التطرق إليه في المثال التالي الخاص بالحالة الخامسة:

الحالة الخامسة:

```
select * from users order by counter limit 1;
```

والـ limit تعني عدد الصفوف المختارة، أي لو قمنا بكتابة المثال السابق بدون الـ limit ستجد أن جميع البيانات سيتم اختيارها، ولكن باستخدام الـ limit نقوم بتحديد عدد الصفوف التي سيتم اختيارها استناداً إلى طريقة ترتيبنا للبيانات، فكما تلاحظ قمنا بترتيب البيانات بحسب الحقل counter ولم نذكر (desc) ولذلك فالبيانات يتم ترتيبها من الأكبر إلى الأصغر، وبالتالي فاختيارنا للحقل الأول يقضي باختيار بيانات الشخص الأكثر كتابة للمواضيع.

بقي أن نذكر طريقي التعديل والحذف ليكتمل الفصل، وسنبدأ بطريقة التعديل على البيانات الموجودة في الجدول users من قاعدة البيانات PHP، والمثال التالي يوضح الطريقة التي سيتم شرحها بعد المثال:

```
update users set
 ،name = “Naser”
counter = 30
where name=”Ahmad”;
```

الجملة update تعني تحديث أو (قم بتحديث)، والـ users هو اسم الجدول الذي نعمل عليه، وفي السطر الثاني قمنا بإسناد القيمة Naser إلى الحقل name، والسطر الذي يليه قمنا بإسناد القيمة ٣٠ إلى الحقل counter، ولكن لو توقفنا هنا بدون ذكر الصف الذي سيتم التعديل عليه، سيتم تعديل كافة الصفوف في الجدول مهما كان عددها، ولذلك كتبنا في النهاية where name=”Ahmad”، بمعنى أن التغيرات السابقة ستحدث فقط على الصف من البيانات التي يحتوي فيها الحقل name على القيمة Ahmad.

ربما يكون المثال غير واضح بشكل كافٍ، ولكن مع التمرس والمحاولة ستجد أن المسألة منطقية وواضحة بشكل كبير، عموماً لم يبق لدينا إلا طريقة الحذف، سواء كان لكل البيانات في الجدول، أو لصف معين من البيانات وسنرى ذلك في المثالين التاليين، وهما ما سنختم به هذا الفصل:

```
delete from users;
```

الأمر السابق كفيل بإلغاء جميع الصفوف في الجدول users كما هو واضح، ولذلك كن متأكداً من أن التجارب التي تقوم بها هي على بيانات غير هامة.

```
delete from users
where id = 1 ;
```

وهذا الحذف سيتم على الصف الذي يتحقق عليه الشرط، وفي هذه الحالة على الصف من البيانات التي يحتوي فيها الحقل id على القيمة ١.

الدوال [Function]:

يوجد في PHP العديد من الدوال التي تقوم بوظيفة معينة (محددة) كذلك توجد إمكانية إنشاء دوال تؤدي وظيفة خاصة وحديثنا هنا عن هذا النوع من الدوال (كيفية إنشاء دوال)

الدالة: تقوم بتنفيذ شئ معين حيث تأخذ (متغيرات – معطيات) ثم تقوم بمعالجة هذه المتغيرات وتخرج قيمة أخرى.

- الشكل العام – التركيب:

```
Function(المعطيات – المتغيرات – البارامتر) اسم الدالة
{
هنا يتم كتابة الكود
Return (المعطيات – المتغيرات – البارامتر) ;
}
```

- تعريف الدالة:

لكي نقوم بتعريف دالة نكتب كلمة function بعدها اسم الدالة وبعد الاسم نكتب المعطيات – المتغيرات بين قوسين.

مثال:

```
<?
Function aa($s)
?>
```

حيث aa هو اسم الدالة ، وبالتأكيد يمكن أن يكون أي اسم.

($s) هو (المتغير – المعطى – البارامتر)، أي اسم من هذه كما تحب أن تسميه.

مع ملاحظة عدم وضع فاصلة منقوطة بعد هذا السطر.

بعد ذلك نقوم بكتابة كود الدالة (عمل الدالة) بين العلامتين { } ، كما يجب أن ننهي الدالة بكلمة return لإعلام الدالة بأن وظيفتها قد انتهت بالإضافة إلى ذكر اسم المتغير المذكور في تعريف الدالة سابقاً..

مثال:

```
<?
Return($s) ;
?>
```

- استخدامات الدالة:

يمكن وضع الدالة في أي مكان في شيفرة php في أولها أو آخرها بمعنى أنه يمكن استدعاء دالة تم تعريفها في أخر الشيفرة أو العكس.

- إظهار نتيجة الدالة (طباعة الدالة):

نستخدم الأمر الخاص بالطباعة echo أو print وبعده طبعاً اسم الدالة..

مثال:

```
<?
echo aa(5);
print aa(5);
?>
```

مثال كامل:

```
<?
//تعريف الدالة
function aa($a)
{
$a=$a*$a*$a*$a;
return($a);
}
//طباعة ناتج الدالة عند إدخال الرقم ٥ فيها
echo aa(5);
?>
```

هذه الدالـة تقـوم بحـساب عـدد مرفـوع لأس أربعـة بمعنـى أن العـدد مـضروب في نفسه أربع مرات اسم الدالة aa وعند طباعة مخرجات الدالـة لـرقم، كتبنـا أمـر الطباعـة قبـل اسـم الدالـة والـرقم المـراد حـساب الأس الرابـع لـه بـين قوسـين (٥) وهكـذا إذا وضعنا أي رقم آخـر سـوف تقـوم الدالـة بحـساب الأس الرابـع للـرقم مباشـر وفي مثالنـا هذا يتم طبع الرقم ٦٢٥.

نقطة أخرى هـي أننـا قمنـا بتمرير قيمة ثابتة إلى الدالة، ولذلك يمكننا أن نمرر للدالة متغير كما في المثال التالي:

```
<?
function as($a)
{
$a=$a*$a*$a*3  ;
return($a) ;
}
$z=10 ;
echo as ($z) ;
?>
```

في هذا المثال تقوم الدالة بضرب العدد في نفسه ثلاث مرات ثم في الـرقم ٣، ونلاحظ أننا مررنا المتغير $z إلى الدالـة as وكتبناهـا جميعهـا في سـطر طباعـة نتيجـة الدالـة بالأمر echo. ولذلك تقوم الدالة في هذا المثال بضرب الرقم ١٠ في نفسه ثلاث مرات ثم في ٣ يكـون النـاتج ٣٠٠٠ ومن ثم يتم طباعـة النـاتج، وبطبيعـة الحـال كلمـا غيرنـا قيمة المتغير اختلفت نتيجة الدالة.

- العمليات الرياضية:

هي نفسها العمليات الـتي درسـتها في المرحلـة الابتدائيـة مـن (جمـع +، طـرح -، ضـرب ❖، قـسمة /) والزائـد علـيهم الـذي لم تدرسـه تقريبـاً هـو بـاقي القـسمة (٪)..

مثال شامل على كل العمليات في الـ **PHP**:

```
<?
$a = 6;
$b=2;
$c= $a +$b;
//سوف نحصل على ناتج الجمع ٨

$c= $a -$b;
//سوف نحصل على ناتج الطرح ٤

$c= $a * $b;
//سوف نحصل على ناتج الضرب ١٢

$c= $a /$b;
//سوف نحصل على ناتج القسمة ٣

$a = 7;
$b=2;
$c= $a % $b;
//سوف نحصل على باقي القسمة ١
?>
```

- عمليات **Assignment**:

(=)

احفظ القيمة في المتغير، بمعنى خزن القيمة ٣ في المتغير $a:

```
<?
$a = 3;
print $a;
//يطبع ٣
?>
```

(+=)

إضافة قيمة إلى قيمة في نفس المتغير: Solomon

```
<?
$a = 3;
$a += 3;
print $a;
//يطبع ٦
?>
```

(-=)

اطرح المقدار واحد من المقدار ثلاثة في المتغير $a:

```
<?
$a = 3;
$a -= 1;
print $a;
//يطبع ٢
?>
```

(❖=)

يضرب القيمة ٣ بالقيمة ٢ ويكون الناتج مخزن في نفس المتغير:

```
<?
$a = 3;
$a *= 2;
print $a;
//يطبع الناتج ٦
?>
```

(/=)

يقسم قيمة على قيمة أخرى:

```
<?
$a = 6;
$a /= 2;
print $a;
//يطبع ناتج القسمة ٣
?>
```

(.=)

دمج سلسلة حرفية:

```
<?
$a = "This is ";
$a.= "a test.";
print $a;
//يطبع الجملة التالية:
// This is a test.
?>
```

- عوامل الإضافة و الطرح:

لو افترضنا أننا لدينا المتغير a=3$ و أردنا إضافة واحد إليه بحيث يصبح ٤ أو طرح واحد منه بحيث يصبح ٢ ، لدينا العوامل التالية:

a$++ أرجع قيمة a ثم أضف واحد إليها

++a$ أضف واحد إليها ثم أرجع القيمة

a$-- أرجع القيمة ثم اطرح واحد منها

--a$ اطرح واحد ثم أرجع القيمة

value++

يتم إضافة واحد إلى الرقم خمسة:

```
<?
$a = 5;
print ++$a;
//يطبع القيمة ٦
?>
```

++value

يرجع القيمة نفسها وفي استخدام ثانٍ تزيد القيمة واحداً:

```
<?
$a = 5;
```

```
print $a++;
//طباعة الرقم ٦
print "<br>";
print $a;
//طباعة الرقم ٥
?>
```

value--

يطرح من القيمة واحداً:

```
<?
$a = 5;
print --$a;
//يطبع الرقم ٤
?>
```

--value

يرجع القيمة نفسها وفي استخدام ثانٍ يطرح منها واحداً:

```
<?
$a = 5;
print $a--;
//يطبع الرقم ٤
print "<br>";
print $a;
//يطبع الرقم ٥
?>
```

- عمليات المقارنة **Comparasion Operators**:

b == a المتغيران متساويان..

b===a المتغيران متساويان و من نفس النوع..

a$ =!b$ المتغير الأول لا يساوي الثاني..

b==! a المتغير الأول لا يساوي الثاني وليس من نفس النوع..

a$ < b$ أكبر من..

a$ < b$ أصغر من..

a$ >=b$ أكبر من أو يساوي..

a$ <=b$ أصغر من أو يساوي..

== (تساوي)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني:

```
<?
$x = 7;
$y = "7";
if ($x == $y) print $x. " تساوي ". $y;
//يطبع ٧ تساوي ٧
?>
```

=== (تساوي ومن نفس النوع)

تساوي القيمة المخزنة في المتغير الأول بالقيمة المخزنة في المتغير الثاني وتكون القيم من نفس النوع (حرفية – عددية):

```
<?
$x = 7;
$y = 7;
if ($x === $y) print $x. " is identical to ". $y;
//يطبع ٧ is identical to 7
?>
```

!= (لا تساوي)

إذا كانت القيم المخزنة في المتغيرين غير متساوية:

```
<?
$x = 8;
$y = 4;
if ($x != $y) print $x. " لا تساوي ". $y;
//يطبع ٨ لا تساوي ٤
?>
```

!== **(لا تساوي ولا من نفس النوع)**

إذا كانت القيم المخزنة في المتغيرين غير متساوية وليست من نفس النوع:

```
<?
$x = 8;
$y = 9;
if ($x !== $y) print $x. " ليست من نفس نوع ". $y;
//يطبع ٨ ليست من نفس نوع ٩
?>
```

> **(أقل من)**

مقارنة بين قيمتين واحدة أقل من الأخرى:

```
<?
$x = 5;
$y = 9;
if ($x < $y) print $x. " أقل من ". $y;
//يطبع ٥ أقل من ٩
?>
```

< **(أكبر من)**

مقارنة بين قيمتين واحدة أكبر من الأخرى:

```
<?
$x = 9 ;
$y = 5;
if ($x > $y) print $x. " اكبر من ". $y;
//يطبع ٩ أكبر من ٥
?>
```

=> **(أقل من ويساوي)**

مقارنة بين قيمتين واحدة أقل من الأخرى أو مساوية لها:

```
<?
$x = 5;
```

```
$y = 5;
if ($x <= $y) print $x;
//يطبع القيمة ٥
?>
```

=< (أكبر من ويساوي)

مقارنة بين قيمتين واحدة أكبر من الأخرى و مساوية لها:

```
<?
$x = 7;
$y = 5;
if ($x >= $y) print $x;
//يطبع القيمة ٧
?>
```

العمليات المنطقية Logical Operations:

لكي تكون قيمة الشرط صحيحة فيجب أن تنطبق القواعد التالية الخاصة بكل عامل منطقي على حدة، والعوامل هي:

(and) يجب تحقق الاثنين $a and $b

(or) يجب تحقق كلاهما أو إحداهما $a or $b

(Xor) يجب تحقق إحداهما وليس كلاهما $a xor $b

(!) نفي تحقق الشرط نفي لقيمة $a !$a

ملاحظة: يمكن كتابة الـ (and) بالشكل التالي (&) والـ (or) بالشكل التالي (|) والـ (Xor) بالشكل التالي (^)..

And (و)

إذا تحقق الشرطان، بمعنى المتغير الأول يساوي ٧ والمتغير الثاني يساوي ٥ نفذ أمر الطباعة واطبع صحيح:

```
<?
$x = 7;
$y = 5;
if (($x == 7) and ($y == 5)) print "صحيح";
//يتم طباعة صحيح
?>
```

Or (أو)

إذا كان أحد الشرطين صحيح أو الاثنين صحيحين نفذ أمر الطباعة:

```
<?
$x = 7;
$y = 5;
if (($x == 7) or ($y == 8)) print "True";
//يطبع True
?>
```

Xor

إذا تحقق أحد الشرطين وليس الاثنين معاً ينفذ أمر الطباعة:

```
<?
$x = 7;
$y = 5;
if (($x == 7) xor ($y == 8)) print "True";
//تحقق شرط واحد فقط فيتم طباعة كلمة True
?>
```

! (النفي)

إذا كانت جملة الشرط غير صحيحة نفذ أمر الطباعة:

```
<?
$y = 5;
if (! ($y == 10)) print "True";
    يطبع True لأن المتغير القيمة المخزنة فيه غير صحيحة
?>
```

&&

المعامل && له نفس وظيفة (and) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات:

```
<?
$x = 7;
$y = 5;
if (($x == 7) && ($y == 5)) print "True";
// Trueيطبع
?>
```

||

المعامل || له نفس وظيفة (or) لكن الاختلاف في ترتيب تنفيذ أولويات العمليات:

```
<?
$x = 7;
$y = 5;
if (($x == 7) || ($y == 5)) print "True";
// Trueيطبع
?>
```

١- الدالة **mysql_connect**:

```
integer mysql_connect(string host، string username، string password);
```

تقوم هذه الدالة بالاتصال مع قاعدة البيانات وتعيد لك رقم يفيدك إذا كان لديك أكثر من اتصال بقواعد البيانات، احتفظ به لاستخدامه في دوال أخرى تالية إذا كان هناك حاجة لذلك كما قلنا، أما الوضع الطبيعي فلا يحتاج إلا إلى الاتصال بالطريقة السابقة فقط وبدون الاحتفاظ بأي رقم، فقط مرر للدالة اسم الخادم واسم المستخدم وكلمة المرور، ولكن يتوجب عليك بعد الانتهاء أن تغلق الاتصال باستخدام الدالة mysql_close

مثال:

```
<?
$link = mysql_connect("db.azzozhsn.f2s.com"،"mag"، "Pass");
?>
```

٢- الدالة **mysql_pconnect**:

```
، string username،integer mysql_pconnect(string host
strin password);
```

هذه الدالة تقوم بما تقوم به الدلة السابقة إلا أنه لا يتوجب عليك إغلاق الاتصال،

مثال:

```
<?
"Pass"); ،"mag"،$link = mysql_pconnect("db.azzozhsn.f2s.com"
?>
```

٣- الدالة **mysql_select_db**:

```
integer link);،boolean mysql_select_db(string database
```

تقوم هذه الدالة باختيار قاعد البيانات المحدد لها. مثال:

```
<?
integer link); ،mysql_select_db(string database
?>
```

٤- الدالة **mysql_db_query**:

```
، string query،boolean mysql_db_query(string database
integer link);
```

تقوم هذه الدالة بتنفيذ سطر SQL على قاعدة البيانات المفتوحة بالمعطى database مثال:

```
<?
"Pass");  ،"mag"،$link = mysql_connect("db.azzozhsn.f2s.com"
$Query = "DELETE FROM magazine";
$link); ، $Query،$result = mysql_db_query("mag"
?>
```

٥- الدالة **mysql_close**:

```
boolean mysql_close(integer link);
```

تقوم هذه الدالة بقطع (إغلاق) قاعدة البيانات، مرر لها رقم الاتصال المعاد من الدالة mysql_connect

مثال:

```
<?
//الاتصال بقاعدة البيانات. .
$link = mysql_connect("localhost"،"mag"، "Pass");
//إغلاق الاتصال بقاعدة البيانات. .
mysql_close($link);
?>
```

٦- الدالة **mysql_query**:

```
integer = mysql_query(string query، integer link);
```

تقوم هذه الدالة بما تقوم به الدالة mysql_db_query تقريباً إلا أن الدالة mysql_query يقتصر عملها على قاعدة البيانات المحددة بالدالة mysql_select_db.

في حالة عدم تمرير رقم الاتصال فستعمل الدالة على الاتصال الأخير.

مثال:

```
<?
$link = mysql_connect("localhost"،"mag"، "Pass");
$query = "DELETE FROM magazine";
$result = mysql_query($query، $link);
?>
```

٧- الدالة **mysql_errno**:

```
integer mysql_errno(integer link);
```

تقوم هذه الدالة بإعادة رقم آخر خطأ حدث في التعامل مع قاعدة البيانات.

٨- الدالة **mysql_error**:

```
string mysql_error(integer link);
```

تعيد هذه الدالة رسالة الخطأ الحاصل في قاعدة البيانات.

٩ - الدالة **mysql_create_db**:

```
boolean mysql_create_db(string databasename، integer link);
```

تقوم هذه الدالة بإنشاء قاعدة بيانات جديدة مرر لها اسم قاعدة البيانات ورقم الاتصال العائد من الدالة mysql_connect أومن الدالة mysql_pconnect..

مثال:

```
<?
//حيث إن الفراغ هو الباسوورد az المتصل بقاعدة بيانات اسمها
$link = mysql_pconnect("localhost"، "az"، "");
//إنشاء قاعدة بيانات جديدة
if (! mysql_create_db($link، "mag"))
{
    print("فشل إنشاء قاعدة البيانات الجديدة ")
    exit();
}
?>
```

١٠- الدالة **mysql_drop_db**:

```
boolean mysql_drop_db(string databasename، integer link);
```

تقوم هذه الدالة بحذف قاعدة البيانات المحددة بالمعطى databasename..

١١- الدالة **mysql_list_dbs**:

```
integer mysql_list_dbs(integer link);
```

تقوم هذه الدالة بإعادة موشر لكل قواعد البيانات الموجودة في الخادم لغرض استعمالها مع الدالة mysql_fetch_row وأمثالها.

١٢- الدالة **mysql_field_seek**:

```
boolean mysql_field_seek(integer result، integer field);
```

تقوم هذه الدالة بتحديد الحقل الممرر إليها رقمه. مثال:

```
<?
//حيث إن الفراغ هو الباسوورد az المتصل بقاعدة بيانات اسمها
$dbLink = mysql_pconnect("localhost"،"az"، "");
Authors //اختيار قاعدة البيانات
mysql_select_db("Authers"، $dbLink);
Adress //اختيار جميع الحقول من الجدول
$Query = "SELECT * FROM adress";
$result = mysql_query($Query، $dbLink);
//الانتقال إلى الحقل الثاني اعتماداً على عملية الاختيار
mysql_field_seek($reslut، 1);
?>
```

١٣- الدالة **mysql_field_name**:

```
string mysql_field_name(integer result، integer feild);
```

تعيد هذه الدالة اسم الحقل المحدد بالرقم الممرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول. مثالها سيأتي بعد قليل.

١٤- الدالة **mysql_field_type**:

```
string mysql_field_type(integer result، integer feild);
```

تعيد هذه الدالة نوع الحقل المحدد بالرقم الممرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول. المثال سيأتي بعد قليل أيضاً..

١٥- الدالة **mysql_field_len**:

```
string mysql_field_len(integer result، integer feild);
```

تعيد هذه الدالة طول الحقل بالبايت المحدد بالرقم المرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول. المثال بعد قليل..

١٦- الدالة mysql_field_flags:

```
integer feild);،string mysql_field_flags(integer result
```

تعيد هذه الدالة وصف الحقل المحدد بالرقم الممرر إليها والذي يبدأ بالرقم صفر للحقل (العمود) الأول.

١٧- الدالة mysql_list:

```
integer link);، string table،mysql_list(string database
```

المثال الشامل:

```
<?
//حيث إن الفراغ هو الباسوورد az المتصل بقاعدة بيانات اسمها
"");  ، "az"،$link = mysql_pconnect("localhost"
//ترتيب الحقول وجلبها
integer link);  ، "table"،$result = mysql_list_field("mag"
//حلقة تكرار للمرور على كل حقل
for ($a = 0; $a < mysql_field_num($result); $a++)
{
$i);  ،    print(mysql_field_name($result
$i));  ،    print(mysql_field_type($result
$i));  ،    print(mysql_field_len($result
i));  ،    print(mysql_field_flags($result
}
?>
```

١٨- الدالة mysql_fetch_field:

```
<?
 integer field);،object mysql_fetch_field(integer result
?>
```

استخدم هذه الدالة لتحصل على معلومات حول حقول الجدول المراد ، الحقول ترقم بدايةً من صفر وصف الحقل مشروح في الجدول التالي:

الخاصة	الوصف
blob	إذا كانت TRUE فالحقل عبارة عن عن حقل بيانات كبير
maxlength	الطول الأقصى للحقل
multiple_key	تكون TRUE إذا كان الحقل مفتاحاً
name	اسم الحقل
not_null	تكون TRUE إذا كان الحقل لا يمكن أن يكون فارغاً
numric	تكون TRUE إذا كان الحقل يرقم تلقائياً
primary_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً رئيساً
unque_key	تكون TRUE إذا كان الحقل يمثل مفتاحاً ثانوياً
zerofill	تكون TRUE إذاكان الحقل يملأ بالقيمة ٠

١٩ - الدالة mysql_fetch_lengths:

```
<?
array mysql_fetch_lengths(integer result);
?>
```

استخدم هذه الدالة لتعيد مصفوفة تحتوي على الطول الأقصى لكل حقل محدد في المعطي result.

```
<?
//Connect to server as azzozhsn no password
"");،"azzozhsn"،$link = mysql_pconnect("localhost"
```

```
//Select th magazine database
$link);،mysql_select_db("magazine"
//Get name and id from magazine
 id FROM magazine';،$Query = 'SELECT name
 $link);،$result = mysql_query($Query
$length = mysql_fetch($result);
//Print length of the third column
print($lengths[2]);
?>
```

٢٠ - الدالة **mysql_fetch_array**:

```
<?
array mysql_fetch_array(integer result);
?>
```

هذه الدالة تعيد مصفوفة تحتوي على قيم سجل وتنقل المؤشر إلى السجل التالي.

مثال:

```
<?
//Connect to server as azzozhsn no password
"");،"azzozhsn"،$link = mysql_pconnect("localhost"
//Select th magazine database
$link);،mysql_select_db("magazine"
//Get name and id from magazine
 id FROM magazine';،$Query = 'SELECT name
 $link);،$result = mysql_query($Query
//Get every row
 MYSQL_ASSOC)){،while($row=mysql_fetch_array($result
 //Print mane and id
    print({$row["id"]}={$row["name"]});
}
?>
```

٢١- الدالة **mysql_fetch_object**:

```
<?
object mysql_fetch_object(integer result)
?>
```

هذه الدالة تشبه الدالة mysql_fetch_array إلا أنها تعيد كائن. عند استدعاء الدالة ينتقل المؤشر إلى السجل التالي في الجدول، وإذا وصل إلى نهاية الجدول ثم استدعيت الدالة مرة أخرى فإنها تعيد القيمة FALSE.

مثال:

```
<?
while($row=mysql_fetch_object(result)){
 //print id and name
 $row->name")،     print ("$row->id
}
?>
```

٢٢- الدالة mysql_fetch_row:

هذه الدالة تعيد مصفوفة تحتوي على قيم حقول سجل من الجدول وكل استدعاء يعيد قيمة الحقول في السجل التالي في الواقع هذه الدالة تشبه الدالتين السابقتين.

مثال:

```
<?
while($row=mysql_fetch_row(result)){
 //print id and name
 $row[1]")،     print ("$row[0]
}
?>
```

٢٣- الدالة mysql_change_user:

```
<?
، string db، string password،musql_change_user(string user
 integer link);
?>
```

استخدم هذه الدالة لتغيير مستخدم قاعدة بيانات المتصل بها. المعطيان db، link اختيارية وفي حالة فقدهما يستعاض عنهما بالاتصال الحالي. هذه الدالة تتطلب إصدار MySQL 3.23.3 أو ما بعدها.

الفصل الثاني عشر
التاريخ في PHP

دالـة التـاريخ في الـ PHP هـي Date ، ولها معـاملين (أي قيمـتين لإعـداد مخرجـات الدالـة)، أحـد المعـاملين إجبـاري والثـاني اختيـاري، أمـا الأول وهـو الأهـم تعتمـد عليـه مخرجـات التـاريخ بشـكل أساسـي مثل ضبط السنة بخـانتين أو ضـبط الـشهر باسـم الشهر.. وغيرها ، أما المعامل الثاني فهو مـا يسمى بـ (UNIX time stamp) وهـو خاص بنظام اليونكس وكيفية تخزين التاريخ فيه، عموماً ما يهمنـا هنـا هـو المعامل الأول وهو ما يسمى بـ (Format String)، وكمثال على ما ذكرنا:

```
<?
$today = date(Y-m-d);
echo $today;
?>
```

هـذا المثـال سيقوم بطباعـة تـاريخ اليـوم علـى الـشكل التـالي ٢٠٠٧-٠٣-١٣، ولأهميـة الرموز التي يمكن استخدامها مع الـ Date سأذكر أهمها:

d رقم اليوم في الشهر على شكل خانتين من ٠١ إلى ٣١.

D اسم اليوم في الأسبوع على شكل ٣ خانات مثل Mon أي الاثنين.

g رقم الساعة في اليوم من ١ إلى ١٢.

j رقم اليوم في الشهر من ١ إلى ٣١ بدون وضع الصفر.

m رقم الشهر في السنة على شكل خانتين من ٠١ إلى ١٢.

y رقم السنة على شكل خانتين، مثلاً ٠٢.

Y رقم السنة على شكل أربع خانات، ومثالها ٢٠٠٧.

هذا من أهم الرموز لكي تتضح الصورة فقط، ولعلنا نتطرق لها بشكل أوسع قريباً.

لتحويل التاريخ إلى اللغة العربية نحتاج أن ننشئ جدولاً في قاعدة البيانات، فلذلك قم بنسخ الكود التالي والصقه في خانة Run SQL query في الـ PHPMyadmin أو بأي طريقة أخرى تراها ، الاهم انشاء الجدول.

```
CREATE TABLE month_name (
id tinyint(4) NOT NULL default '0',
month text NOT NULL
) TYPE=MyISAM;

INSERT INTO month_name VALUES (1, 'كانون الثاني');
INSERT INTO month_name VALUES (2, 'شباط');
INSERT INTO month_name VALUES (3, 'أذار');
INSERT INTO month_name VALUES (4, 'نيسان');
INSERT INTO month_name VALUES (5, 'أيار');
INSERT INTO month_name VALUES (6, 'حزيران');
INSERT INTO month_name VALUES (7, 'تموز');
INSERT INTO month_name VALUES (8, 'أب');
INSERT INTO month_name VALUES (9, 'أيلول');
INSERT INTO month_name VALUES (10, 'تشرين الأول');
INSERT INTO month_name VALUES (11, 'تشرين الثاني');
INSERT INTO month_name VALUES (12, 'كانون الأول');
```

بعد انشاء هذا الجدول يجب أن يكون لديك جدول آخر يحتوي على التاريخ المراد تحويليه، ولنفترض أن لديك الجدول (news) يحتوي على الحقول (title، date) ويحتوي على البيانات التالية:

date title

٢٠٠٧-٠٤-٢٠ الخبر الأول

٢٠٠٧-٠٤-٢٥ الخبر الثاني

٢٠٠٧-٠٥-٠١ الخبر الثالث

قم بإنشاء الجدول:

```
CREATE TABLE news (
title text NOT NULL ,
date date NOT NULL default '0000-00-00'
) TYPE=MyISAM;
```

```
INSERT INTO news VALUES ('الخبر الأول'،'٢٠٠٧-٠٤-٢٠');
INSERT INTO news VALUES ('الخبر الثاني'،'٢٠٠٧-٠٤-٢٥');
INSERT INTO news VALUES ('الخبر الثالث'،'٢٠٠٧-٠٥-٠١');
```

بقـي أن نقـوم بتحويـل التـاريخ إلى العربيـة، وإدراجـه في صـفحة PHP، ولعمـل ذلـك سنقوم باستخدام دالـة تسمى Date_Format مـن خـلال طلـب لقاعـدة البيانـات، نحدد من خلاله طريقة جلب البيانات ووضعها بالصورة المطلوبة.

بقي أن نذكـر أننـا سـوف نضطر إلى كتابـة طلبين لقاعـدة البيانـات أحـدهما لجلب حقول العنوان (title) والآخر لجلب حقول التاريخ (date) كما يلي:

```
<?
$result = mysql_query("select * from news");
$sql = "SELECT CONCAT(DATE_FORMAT(date،'%d')، \"
\"،month_name.month،\" \"،DATE_FORMAT(date، '%Y'))
AS date FROM news، month_name
WHERE month_name.id = month(date)";
$result2 = mysql_query("$sql");
while ($row=mysql_fetch_array($result)
and  $row2=mysql_fetch_array($result2))
{
$title = $row["title"];
$date = $row2["date"];
echo "$title، $date<br>";
}
?>
```

عند تنفيذ السكريبت، سترى ما يلي:

الخبر الأول، ٢٠ نيسان ٢٠٠٧ .

الخبر الثاني، ٢٥ نيسان ٢٠٠٧ .

الخبر الثالث، ٠١ أيار ٢٠٠٧.

في حالات كثيرة تكون كتابة السكريبت السابق بهذا الشكل مسببة للكثير من المشاكل، و خاصة عند طلب ترتيب للجدول على حسب حقل معين، وهذه المشاكل هي في توافق البيانات مع بعضها البعض، فلو افترضنا في مثالنا السابق أن الخبر الأول الذي يحمل التاريخ ٢٠٠٧-٠٤-٢٠ كان باسم آخر، مثلاً (العنوان الأول)، وبعد إضافة حقول ترتيب لجلب البيانات كالتالي:

```
<?
$result = mysql_query("select * from news
order by title");
\" ،'%d')،$sql = "SELECT CONCAT(DATE_FORMAT(date
'%Y')) ،DATE_FORMAT(date،\" \"،month_name.month،\"
month_name ،AS date FROM news
WHERE month_name.id = month(date)";
$result2 = mysql_query("$sql");
while ($row=mysql_fetch_array($result)
and  $row2=mysql_fetch_array($result2))
{
$title = $row["title"];
$date = $row2["date"];
 $date<br>"; ،echo "$title
}
?>
```

ستجد أن النتائج هي:

الخبر الثالث، ٢٠ نيسان ٢٠٠٧

الخبر الثاني، ٢٥ نيسان ٢٠٠٧

العنوان الأول، ٠١ أيار ٢٠٠٧ وهذه بطبيعة الحال مشكلة في توافق البيانات.

ولحلها يجب أن نوافق بين الطلبين لقاعدة البيانات، بمعنى أنه إذا رتبنا الطلب الأول حسب (title) يجب أن نفعل ذلك مع الطلب الثاني بتعديله ليصبح:

```
<?
\" ،'%d')،$sql = "SELECT CONCAT(DATE_FORMAT(date
'%Y')) ،DATE_FORMAT(date،\" \"،month_name.month،\"
month_name ،AS date FROM news
WHERE month_name.id = month(date)
order by title";
 ?>
```

وبالتالي تصبح البيانات المخرجه كالتالي:

الخبر الثالث، ٠١ أيار ٢٠٠٧

الخبر الثاني، ٢٥ نيسان ٢٠٠٧

العنوان الأول، ٢٠ نيسان ٢٠٠٧

وهي بالتأكيد صحيحة.

بعد المقدمات السابقة والهامة في معرفة أساسيات اللغة يمكننا بداية كتابة البرامج بلغة الـ PHP، وبطبيعة الحال سنبدأ من أصغر الأساسيات وأهمها في كتابة البرامج عموماً وهي المتغيرات.

المتغيرات في لغة الـ PHP تبدأ بعلامة الدولار ($)، ولاسناد قيمة لذلك المتغير نستخدم علامة المساواة (=)، فرضاً لدينا المتغير (Name) والقيمة (Khaled) فنكتب ما يلي:

```
<?
$Name = "Khaled";
?>
```

هذا في حالة المتغيرات النصية (Text)، وفي حالة المتغيرات الرقمية (Numbers) يمكن تعريف متغير (Counter) الذي يحمل القيمة (١٧) كالتالي:

```
<?
$Counter = 17;
?>
```

الفرق الواضح في طريقة تعريف المتغيرين النصي والرقمي هو عدم وجود علامات التنصيص في تعريف المتغيرات الرقمية بينما يجب وضع علامات التنصيص في تعريف المتغيرات النصية.

نقاط هامة في تسمية المتغيرات:

- اسماء المتغيرات في كثير من لغات البرمجة لا تتعدى ٢٥٥ حرف (المقصود بها الخانات سواء كانت حروف أو أرقام أو علامات أخرى)، و في لغة الـ PHP لا يوجد حدود على عدد الخانات في تسمية المتغيرات، ولكن في الغالب لن تحتاج إلى أكثر من ١٥ خانة لتسمية أي متغير، لأن المبالغة في تسمية المتغيرات تسبب مشاكل في تذكر المتغيرات وما تحتوية من قيم.

- بداية كل متغير يجب أن يبدأ بحرف (يعني حرف هجائي) أو علامة (_) Underscore، مع تجاهل علامة الـ $ لأنها لا تحسب من اسم المتغير.

- يمكن أن يحتوي اسم المتغير على الحروف أو الأرقام أو علامة (_) فقط، أما العلامات الأخرى مثل (+، -، ❖، /) أو الـ & لا يمكن كتابتها في اسم المتغير.

- المتغير ($Name) يختلف عن المتغير ($name) لاختلاف حالة حرف الـ N، ولذلك يجب التأكد من اسم المتغيرات بدقة لتجنب حدوث مشاكل في الوصول إلى متغير معين، وبالتأكيد لو كان لديك أسلوب خاص في تسمية المتغيرات لسهولة الوصول إليها وتذكرها ستكون كتابة السكريبتات أسهل بكثير.

- يستحسن أن تكون أسماء المتغيرات دالة على معانيها، بمعنى أنه لمتغير مثل عداد الزوار يستحسن أن يكون ($counter)، ولمتغير مثل اسم المستخدم ($user).. الخ.

التعامل مع المتغيرات:

فائدة المتغيرات تكمن في طريقة استخدامها في كتابة السكريبت، وكما ذكرنا سابقاً إنه لطباعة متغير معين نستخدم أمر الطباعة (echo) أو (print) كما يلي:

```
<?
$name = "Naser";
echo $name;
?>
```

في البداية سيتم إسناد القيمة (Naser) إلى المتغير ($name)، وفي السطر الثاني تتم طباعة المتغير، أو بالأحرى القيمة المسندة إلى المتغير.

أنواع البيانات [Data Types]:

في الأمثلة السابقة قمنا بإسناد قيمتين عددية ونصية إلى متغيرين، وبينا الفرق بينهما، وفي لغة الـ PHP بشكل عام يوجد أكثر من هذين النوعين من البيانات، سأشرح بعضاً منها الآن، والبقية في الفصول القادمة:

- البيانات النصية (String).
- البيانات العددية الصحيحة (Integer).
- البيانات العددية الكسرية (Double).
- المصفوفات (Array).
- الكائنات (Object).
- البيانات الغير معروفة !.

البيانات النصية [String]:

هـي البيانـات الـتي تكـون بـين علامـات التنـصيص " " بغـض النظـر عـن محتوهـا ، فيمكن أن تكـون حـروف أو أعـداد أو رمـوز أو غيرهـا ، ومثـال ذلـك كمـا ذكرنـا سـابقاً :

```
<?
$user = "Khaled";
$age = "13.5";
?>
```

التعامل مع البيانات النصية [String]:

لإضافة المتغيرات التي تحتوي على بيانات نصية مع متغيرات من نفس النوع نحتاج إلى عملية دمج بين المتغيرات، ولعمل ذلك نكتب:

```
<?
$total = $user. $age;
?>
```

في هـذه الحالـة سـيتم إسـناد القيمـة Khaled13.5 إلى المـتغير ($total).

إذا أردنا وضع مسافة بين المتغيرين نضيف متغير جديد يحتوي على المسافة وهو ($space) ثم نقوم بعملية الدمج كالتالي:

```
<?
$space = " ";
$total = $user. $space. $age;
?>
```

وفي هذه الحالة سيتم وضع القيمة Khaled 13.5 في المتغير ($total)، وبطبيعة الحال يمكن استخدم المتغيرات النصية داخل متغيرات نصية أخرى، حيث سيتم تعويض المتغير بقيمته الأصلية.

البيانات العددية [Numeric]:

وكما ذكرنا في التقسيم السابق أنها نوعين (الأعداد الصحيحة Integer) و (الأعداد الكسرية Double)، وكمثال على النوعين:

```
<?
$integer1 = 233;
$integer2 = -29
$double1 = 5.27
$double2 = -4.6
?>
```

التعامل مع البيانات العددية [Numeric]:

العمليات الحسابية المشهورة (+، -، *، /) بالاضافة إلى باقي القسمة (٪) عمليات شائعة جداً في التعامل مع المتغيرات العددية، وبطبيعة الحال لن نحتاج إلى ذكر أي مثال عن هذه العمليات، وسنكتفي بذكر بعض النقاط الأساسية التي قل ما يخلو سكريبت منها.

أول النقاط هي إضافة المتغير إلى نفسه، بمعنى تعريف عملية حسابية على متغير معين بحيث تخزن القيمة في نفس المتغير، مثلاً لو كان لديك عدد الزوار وتريد في كل مرة أن يزيد عدد الزوار بـ ١، يمكنك كتابة ما يلي:

```
<?
$counter = $counter + 1;
?>
```

بالتالي ستتم زيادة المتغير ($counter) بـ ١ في كل مرة يتم فيها تنفيذ السكريبت، وبطريقة أخرى يمكن كتابة السطر السابق كالتالي:

```
<?
$ccounter = $counter++;
?>
```

والـ ++ تعني زيادة قدررها (١) على قيمة المتغير الأصلية، وكذلك الـ -- تعني طرح ١ من القيمة الأصلية.

وفي حالة الرغبة بزيادة أي عدد أخر (غير الواحد) على أي متغير بأسلوب الطريقة الثانية يمكن كتابة ما يلي:

```
<?
$counter +=4;
?>
```

وهـذا يعـني زيـادة مقـدارها ٤ علـى قيمـة المـتغير الأصـلية، وبالـسالب كـذلك بـنفس الأسلوب.

ترتيب إنجاز العمليات الحسابية:

يوجد بعض الرمـوز والعمليـات الـتي تـسبق غيرهـا عنـد البـدء في انجـاز عمليـة حـسابية معينة، والترتيب المستخدم في الـ PHP كـالتالي:

] -

(int) (double) (string) (array) (object) -- ++ ~ ! -

٪ / ❖ -

.- + -

<< >> -

> => < =< -

== =! === ==! -

& -

| -

&& -

|| -

؟ :-

= =+ =❖ =/ =.=٪ =& =^ =~ =>> =<< -

print -

AND -

XOR -

OR -

، -

بالتأكـيد القائمـة طويلـة وفيهـا تفاصـيل كـثيرة، ولكـن مـن المهـم معرفـة طريقـة إنجـاز العمليـات الحسابية المختلفة لسهولة اكتشاف الأخطـاء ومعرفـة الطريقـة الـصحيحة لكتابـة بعـض العمليـات المعقدة للحصول على ناتج صحيح.

بعض الدوال الهامة في التعامل مع المتغيرات:

- isset: وهي دالة للتأكد من وجود متغير معين، فمثلاً:

```
<?
echo isset($age);
?>
```

ستتم طباعة الرقم ١ إذا كان المتغير ($age) موجوداً (تم إنشائه مسباقاً)، والعكس إذا كان غير موجود ستتم طباعة الرقم ٠، وهذه الدالة يتم استخدامها كثيراً في الشروط وهذا ما سنتطرق إليه لاحقاً.

- unset: هذه الدالة تعمل على مسح المتغير من الذاكرة كلياً، فقط قم بعمل التالي:

```
<?
unset($age);
?>
```

وفي هذه الحالة سيتم مسح المتغير ($age) بشكل كامل.

- empty: وهذه الدالة معاكسة للدالة isset بحيث لو كتبنا ما يلي:

```
<?
echo empty($age);
?>
```

ستتم طباعة الرقم (١) في حالة عدم وجود المتغير ($age) أو أن قيمة المتغير تساوي (٠) و(فراغ)، وفي حالة وجود المتغير ($age) لن تتم طباعة أي شيء.

للتحكم في المواقع عن طريق الـ Session أو الجلسات كما اصطلح على تسميتها، ففي البداية سنتعرف على الـ Session وعن التحكم فيها، ومن ثم استخداماتها بالإضافة إلى بعض الأمثلة، وفي النهاية سنتطرق إلى بعض الأخطاء في كتابة الـ Session وحلول تلك الأخطاء، وفي الفصل القادم بإذن الله تعالى سنتطرق إلى مثال كامل للوحة تحكم مبسطة تتعامل بالـ Session، والأمل أن يكون في هذا الشرح المبسط فائدة للجميع..

- مقدمة عن الـ **Session**:

عند الانتقال من صفحة إلى أخرى في موقع معين فإن بروتوكول الـ HTTP لا يمكنه معرفة أن تلك الصفحات قد تم تصفحها من قبل نفس الشخص، ولكن مع الـ cookies وما نحن بصدده هنا الـ Session تقدم تلك الطريقة، ولذلك وببساطة فإن الـ Session هي مكان على جهاز المتصفح يمكن من خلاله تخزين قيمة معينة للرجوع إليها في حال قام نفس الشخص بالانتقال من صفحة إلى أخرى، ولعل هذا التعريف يصف ببساطة معناها العام ولا يعني ذلك أنه تعريف شامل لكل المعاني..

إذا التعرف على الشخص الذي يقوم بتصفح الموقع هو الهدف الرئيسي للـ Session أو الجلسات، ولكن كيف يتم ذلك، وما هي النقاط الرئيسية التي يجب معرفتها لفهم طريقة التعامل مع الـ Session ؟

أول تلك النقاط أن عملية تسجيل المتغير على جهاز المستخدم له مدة معينة تنتهي بانتهاء الجلسة، ومن هنا جاءت التسمية، أما ما تعنيه الجلسة فهي مصطلح لقيامك بالتصفح من الموقع ومن ثم إغلاق الموقع، ببساطة كل مرة تقوم بزيارة الموقع تبدأ جلسة أو Session جديدة، مع ملاحظة أن هناك طرق للتحكم بوقت الانتهاء كما في الـ cookies، بالإضافة إلى طرق أخرى عن طريق قواعد البيانات وهو حديث سابق لأوانه.

بالنسبة للنقطة الأخرى التي يجب وضعها في الحسبان هي ما يسمى بالـ Session ID أو اختصاراً SID ويعني ذلك (رقم الجلسة)، وهو رقم عشوائي فريد يصعب تكراره أو فلنقل إنه مستحيل لاحتوائه على أرقام وأحرف كبيرة وصغيرة في متغير طويل نسبياً، وهذه القيمة هي الأهم في ما ذكرت، لأنها القيمة الوحيدة التي تربط ما يسمى بالـ Session Variables أو (متغيرات الجلسة) مع جهاز المستخدم، فالـ SID هي القيمة الوحيدة التي يتم تخزينها في جهاز المستخدم (Client)، أما الـ

متغيرات الجلسة Session Variables يتم تخزينها في السيرفر (Server)، فعند التحقق منه وجود هذه القيمة على جهاز المستخدم يمكن الدخول إلى المتغير الآخر المترابط به والمسمى بالـ Session Variable.

النقطة الثالثة هي طريقة التخزين للـ SID و الـ Variables Session، أما الـ SID وكما قلنا إنها تخزن على جهاز العميل (Client) إما عن طريق الـ cookies والتي لها سلبياتها المتعددة أو عن طريق تمريرها عبر الـ HTTP، أما بالنسبة للـ Session Variables فيتم تخزينها في ملفات فارغة على جهاز الـ Server وكذلك في مستويات متقدمة يمكن التحكم بها وتخزينها في قواعد بيانات.

- إعدادت الـ Session:

عن طريق ملف الـ php.ini والذي يحتوي على إعدادت الـ PHP يمكن التحكم بإعدادات الـ Session، وكاستعراض لتلك النقاط المتحكمة بالـ Session سنتطرق أهم النقاط ومعانيها، وللوصول إلى ما نحن بصدده تذكر أن ملف الـ php.ini يوجد في دليل الـ Windows، وللوصول إلى خصائص الـ Session ابحث عنه كلمة (Session) وستجد السطر التالي:

[Session]

من هنا تستطيع التحكم بخيارات الـ Sessions، وكما يظهر في الجدول التالي وصف لأهم الخيارات..

١- الخيار **Session.auto_start**:

بداية تلقائية للـ Session (دون الحاجة لعمل ذلك يدوياً عن طريق Session_start).

٢- الخيار **Session.cache_expire**:

وقت انتهاء الجلسة بالدقائق.

٣- الخيار **Session.cookie_lifetime**:

وقت انتهاء الـ cookie المرتبطة بالجلسة، وهي افتراضياً ستكون ٠ أي أن الـ cookie ستنتهي فترتها مع إقفال الشخص المتصفح للموقع.

٤- الخيار **Session_name**:

اسم الـ Session التي ستستخدم كـ cookie وافتراضياً ستكون PHPSESSID.

٥- الخيار **session.save_path**:

هذا السطر يعني مكان تخزين ملفات الـ Session في جهازك باعتباره سيرفر، وهنا تستطيع أن تضع أي عنوان في جهازك، أما تركه فارغاً فيعني عدم تفعيل الـ Session لديك، بالنسبة لي أقترح أن يكون المجلد Temp داخل الـ Windows هو الدليل الأمثل لاحتوائه على ملفات مؤقتة يمكن حذفها، إذاً العنوان سيكون c:\windows\Temp

هذه في نظري أهم الخيارات التي يجب فهمها.

- بداية الـ Session:

قبل أن تستخدم أياً من دوال الـ Session يجب إخبار السكريبت أن يبدأ جلسة Session، والطريقة هي أن تضع في بداية السكريبت وفي أول سطر فيه بعد علامة الفتح ما يلي:

```
<?
session_start();
?>
```

في هذه الحالة فقط يمكن أن تقوم باستخدام دوال الـ Session الأخرى، أما إذا لم تتم كتابة هذا السطر فلن يتم ذلك.

ملاحظة مهمة حول عملية بداية الـ Session وهي أن تتأكد من أن هذا السطر لا يسبقه عملية إخراج مخرجات، بمعنى آخر أي استخدام لدوال مثل echo أو print، وكذلك لا يسبق هذا السطر أي فراغ وتأكد من هذه النقطة جيداً لأنها كثيرة الحدوث وتعطى الخطأ التالي:

وأسلم طريقة من وجهة نظري أن تضع هذا السطر في بداية ملف الـ header لأنك سنقوم بإدراج هذه الصفحة في كل الصفحات الأخرى وبالتالي يكون السطر هو الأول في كل الحالات..

- تخزين متغيرات الجلسات:

وهي ما نسميها بالـ Session Variables، ولعمل ذلك يوجد لدينا الدالة الواردة في المثال التالي:

```
<?
$user = "AbdulAziz";
session_register("user");
?>
```

ما قمنا بعمله هو التالي:

١- عرفنا متغيراً هو **user** يحتوي على قيمة حرفية.

٢- قمنا بتسجيل هذا المتغير في متغير جلسة (Session Variable) وبنفس الاسم **user** ولكن بدون علامة **$**.

- التعامل مع متغيرات الجلسة:

بعد تسجيل المتغير، يمكن الرجوع إليه بعدة طرق تعتمد على الخيار **register_globals** في ملف الـ **php.ini**، أما إذا كان **on** وهذا هو الاختيار الافتراضي فإن المتغير الذي تم تسجيله في الـ Session يمكن الرجوع إليه كأي متغير آخر، عن طريق اسم المتغير فقط، وفي مثالنا الحالي سيكون **user$**، أما إذا كان الخيار غير مفعل وليس بالصورة التي ذكرتها فيمكن

الرجوع إلى المتغير عن طريق الأمر $HTTP_SESSION_VARs["user"].

أيضاً كنقطة مهمة يجب معرفتها وهي طريقة التحقق من أن متغيراً معيناً قد تم تسجيله أم لا ، وهذه الطريقة مفيدة في الصفحات التي يجب أن يكون فيها المستخدم قد سجل الدخول وبالفعل تمت عملية تسجيل الـ Session له ، في المثال التالي تلك الطريقة:

```
<?
if (session_is_registered("user")) {
echo "أهلاً وسهلاً بكم في السلسلة المعلوماتية الميسرة";
}
else {
echo "لا يسمح لك بالدخول.";
}
?>
```

في هذا المثال سيتم عرض الجملة (أهلاً وسهلاً بكم في السلسلة المعلوماتية الميسرة) إذا كانت عملية تسجيل الـ Session تمت للمتغير **user** ، وسيتم عرض الجملة (لا يسمح لك بالدخول..) في حالة عدم تسجيل الـ Session.

نقطة أخيرة في التعامل مع متغيرات الجلسة ، وهي عملية الغاء تسجيل الـ Session لمتغير معين، وهذه الطريقة تتم عن طريق الدوال **session_unregister** و **session_unset** و **session_destroy** ، أما الفرق بينهم فهو أن الدالة الأولى تقوم بعملية إلغاء التسجيل لـ Session معينة ، أي بتمرير اسم المتغير لها كما في المثال التالي:

```
<?
session_unregister("user");
?>
```

إذاً سيتم إلغاء تسجيل الـ Session المتعلقة بالمتغير **user** فقط، أما الدالة الثانية فستقوم بإلغاء تسجيل جميع الـ Session التي تم تسجيلها من قبل، وفي النهاية

يجب استخدام الدالة الثالثة **session_destroy** لإلغاء الـ SID والانتهاء من التعامل مع الـ Session.

- مثال بسيط عن الـ Session:

سأتطرق إلى مثال بسيط جداً لتوضيح كيفية عمل الـ Session، في البداية قم بوضع الكود التالي في ملف وقم بتسميته phpex1.php:

```
<?
$age = 12;
session_register("age");
echo "$age تحتوي على القيمة age الجلسة<br>";
echo "<a href=phpex2.php>التالي.</a>";
?>
```

الصفحة الثانية احفظها باسم phpex2.php، وضع الكود التالي فيها:

```
<?
echo "أنت في الصفحة الثانية<br>";
echo "$age تحتوي على القيمة age الجلسة<br>";
session_unregister("age");
echo "<a href=phpex3.php>التالي.</a>";
?>
```

الصفحة الثالثة تحتوي على الكود التالي، واسمها phpex3.php:

```
<?
echo "أنت في الصفحة الثالثة<br>";
 echo "$age تحتوي على القيمة age الجلسة<br>";
?>
```

ابدأ من الصفحة الأولى ومن ثم انتقل من صفحة إلى أخرى، حتى تصل إلى الثالثة، بافتراض أنك قمت بتجربة المثال، ستلاحظ أن الصفحة الأولى ستتم طباعة الـ Session التي تم تسجيلها وهي **age** وستظهر القيمة ١٢ في الجملة الطويلة التي تبين أن المتغير **age** يحتوي على قيمة معينة، وفي الصفحة الثانية ستلاحظ نفس الجملة ونفس القيمة تمت طباعتهما، أما في الصفحة الثالثة والأخيرة فتمت طباعة الجملة، لكن الاختلاف أن القيمة ١٢ في متغير الـ Session **age** لم تتم طباعتها، لماذا ؟

لسبب بسيط وهو أننا في الصفحة السابقة قمنا بالغاء تسجيل الـ Session للمتغير **age** وبالتالي فإن الصفحة الثالثة لم تتعرف على متغير مباشر له الاسم **age** ولا على متغير الـ **age Session** ، وبالتالي تمت طباعة الجملة بدون القيمة.

الفصل الثالث عشر
التعامل مع الملفات والمجلدات

كل مبرمج يجب أن يتعامل مع الملفات والمجلدات في بعض النقاط، برنامجك سوف يستخدم الملفات لكي يقوم بتخزين معلومات الإعداد للسكريبت، أو يقوم بتخزين البيانات لقراءتها وكتابتها، أو لكي يقوم بحفظ البيانات المؤقتة، وكمثال فإن أتفه برنامج عداد يحتاج إلى ملف يقوم بتخزين آخر رقم تم الوصول إليه..

الملف: ليس عبارة عن أكثر من بايتات متسلسلة يتم تخزينها على القرص الصلب أو أي مادة تخزينية أخرى.
والمجلد: هو عبارة عن نوع محدد من الملفات يحتفظ بأسماء ملفات أخرى ومجلدات أخرى (تسمى بالمجلدات الفرعية)، كل ما تحتاجه لتتعامل مع الملفات والمجلدات هو كيف يمكنك ربط سكريبتك بهم..

هذا الفصل سيأخذك في جولة لتعلم التعامل مع الملفات والمجلدات وفي نفس الوقت يوفر لك مرجعية لبعض الدوال التي تساعدك في ذلك مما يجعل مهمتك أسهل...

سيقوم هذا الفصل بتغطيه المواضيع التالية:

١ فتح وإغلاق الملف.
٢ القراءة من الملف والكتابة فيه.
٣ مسح وإعادة تسمية الملفات
٤ استعراض وتجوال في الملف.
٥ فتح وإغلاق المجلدات.
٦ نسخ وإعادة تسمية المجلدات.

ملاحظة:

قبل أن نبدأ دعنا ننبهك أن التعامل مع الملفات يختلف من نظام تشغيل إلى آخر ففي أنظمة اليونكس تستخدم المسارات العلامة المائلة للأمام.

مثال

/home/usr/bin/data.txt

بينما في الويندوز فإن المسار يكون كالتالي:

C:\usr\bin\perl

وإذا استخدمنا العلامة الأمامية في PHP للويندوز فإنه يقوم بتحويلها بشكل تلقائي إلى علامة خلفية بينما إذا أردنا استخدام العلامة الأمامية فإننا يجب أن نقوم بتكرار العلامة لكي يتم التعرف عليها.

PHP\C:\\windows\

التعامل مع الملفات

يوفر الـPHP نوعين من الدوال المتعلقة بالملفات فهناك نوع من الدوال يستخدم مقبض للملف (file handle) أو ما يسمونه بالمؤشر (pointer) في العادة، بينما بعض الدوال تستخدم قيمة حرفية تشير إلى موضع الملف مباشرة...

مقبض الملف ليس أكثر من عدد صحيح (integer) يقوم بتعريف الملف المراد فتحه حتى يتم إغلاقه، إذا كان هناك أكثر من ملف مفتوح فإن لكل ملف مقبضه التعريفي الخاص به، وبالطبع فإنه لا يتوجب عليك معرفة هذا الرقم....

على سبيل المثال فإن الدالة fwrite() تقوم بفتح الملف لكتابة بيانات إليه وهي تستخدم مقبض لكي تقوم بالتعرف إلى الملف وفتحه..

```
'Hello World'); ،Fwrite ($fp
```

بينما الدالة file() التي تستخدم للقراءة من الملف تقوم باستخدام قيمة نصية تقوم بالإشارة إلى مكان الملف بشكل مباشر لكي يتم التعامل معه..

لا تصب بالرعب والخوف من هذا الكلام فأنا أعلم أنه قد يكون غامضاً عليك..
تنفس الصعداء وجهز لنفسك كأساً من الشاي لأننا سنبدأ في الجد الآن....

ملاحظة: ستجد أن أغلب الدوال أو معظمها أو كلها تقريباً تقوم بإرجاع القيمة True إذا تمت بنجاح والقيمة False إذا فشلت في الحصول على هدفها..
لنبدأ الآن مع سكريبتات مبسطة للعمل مع الملفات..

فتح وإغلاق الملفات

Fopen:

تستخدم هذه الدالة ثلاثة عوامل هي مسار الملف (path) والوضع له (للقراءة، للكتابة.....) بالإضافة إلى مسار الـInclude فيه وتقوم هذه الدالة بإرجاع مقبض للملف...

قد تواجهنا مشاكل أحياناً فقد يكون الملف غير منشأ أو أننا لا نملك صلاحيات عليه ولذلك فإنه يمكننا اختبار القيمة التي ترجعها هذه الدالة فإذا كانت القيمة صفراً فهذا معناه أن الدالة فشلت في إرجاع مقبض الملف أو نوعه، أما إذا كانت القيمة هي واحد فهذا معناه أن الدالة قد نجحت في فتح الملف.

مثال

```
$fp=fopen (“./data.txt”، “r”);
if (!$fp) die (“فشل في قراءة الملف تأكد من التراخيص ومن مسار الملف”);
```

يمكننا كتابة المثال أيضاً بالشكل التالي:

```
If (!($fp=fopen (“./data.txt”، “r”))) die (“لا يمكن القراءة من الملف”);
```

لاحظ أننا قلنا سابقاً إن هناك دوال تستخدم للتعامل مع الملفات تستخدم مقبض وهذا المقبض هو عبارة عن رقم، في مثالنا هذا يتحدد رقم المقبض الذي هو المتغير $fp الذي يخزن فيه مكان الملف وما إذا كان قابلاً للفتح أو لا أو يعمل أو لا يعمل، والنتيجة التي تتخزن في المتغير $fp هي رقم مثلما قلنا سابقاً وهو صفر إذا كان الملف لا يعمل أو واحد إذا تم فتح الملف بنجاح..

الآن دعنا نناقش معاملات الدالة fopen التي تقوم بإعطائنا رقم المقبض..

أول معامل هو مسار الملف على القرص الصلب

لنفرض أن لديك مجلداً قمت بإنشائه في مجلد السكريبتات الرئيسي لديك الذي يسمى htdocs وأسميته data

ولنفرض أن سكريبتك يستخدم ملفين:

١- ملف للقراءة والكتابة يسمي data.txt.

٢- وملف يقوم بعرض المدخلات والإضافة إليها اسمه script.txt.

حسناً لدينا الآن ثلاث حالات للسكريبت

الحالة الأولى:

أن يكون الملفان في نفس المجلد (data) وعند ذلك يمكنك فتح الملف الذي تريد فتحه بذكر اسمه فقط من غير إضافات.

```
$fp=fopen (“data.txt”، “r”);
```

الحالة الثانية:

أن يكون هناك مجلد في نفس مجلد الـdata باسم آخر ولنقل أن هذا الاسم هو gb وفيه ملف data.txt على ذلك فإننا نكتب المسار المطلق لهذا المجلد كالتالي:

```
$fp=fopen ("./gb/data.txt"، "r");
```

الحالة الثالثة:

أن يكون الملف الذي تريد قراءته موجود في المجلد htdocs بينما السكريبت موجود في المجلد data الموجود داخل htdocs على ذلك نكتب المسار النسبي كالتالي:

```
$fp=fopen ("../data.txt"، "r");
```

لاحظ النقطة التي تسبق العلامة الأمامية جيداً..

أتمنى أن تكون فهمت من هذا الكلام ما هو المقصود بالمسار المطلق والمسار النسبي..

يمكننا أيضاً وضع رابط صفحة في موقع آخر ولكننا لن نستطيع الكتابة عليه بل قراءته فقط.

مثال:

```
If (!($fp=fopen ("http://www.swalif.net/softs/index.php"، "r")))
die ("لا يمكن القراءة من الملف");
```

ينقصنا نقطة يجب أن نتكلم عنها وهي عند تحديد العامل use_include_path

العامل الثاني الذي نستخدمه للملفات هو حالة الملف:

(للقراءة، للكتابة، للإضافة إليه) يحدد وضعية الملف حال فتحه إذا كان للقراءة فقط أو للكتابة فقط أو للاثنين معاً أو للإضافة، وأرتبها هنا في جدول بسيط..

القيمة	الوصف
r	تفتح الملف للقراءة فقط ويكون المؤشر في بداية الملف
r+	يفتح الملف للقراءة والكتابة ويضع المؤشر في بداية الملف
w	يفتح الملف للقراءة فقط، أي بيانات موجودة سيتم مسحها، إذا لم يكن الملف موجوداً سيحاول PHP إنشاؤه.
w+	يفتح الملف للقراءة والكتابة، أي بيانات موجودة سيتم مسحها، إذا لم يكن الملف موجوداً سيحاول PHP إنشاؤه.
a	يفتح الملف للإضافة فقط، إذا لم يكن الملف موجوداً سيحاول PHP إنشاؤه، سيكون المؤشر في نهاية الملف.
a+	يفتح الملف للقراءة و للإضافة، إذا لم يكن الملف موجوداً سيحاول PHP إنشاؤه، سيكون المؤشر في نهاية الملف.
b	يستخدم لفتح وقراءة ملفات الصور على نظام أو سيرفرات الويندوز فقط.. أما الينوكس فالعوامل السابقة تتعامل مع ملفات الصور بشكل عادي..

هناك مؤشر للملفات يحدد إذا ما كنت ستكتب من نهاية أو بداية الملف أو حتى من وسطه أو من أي مكان بالملف، ستعرف كيفية التحكم بهذا المؤشر بعد قليل.

العامل الثالث هو تحديد ***use_include_path***

فـإذا قمـت بتحديـد قيمتـه إلى (١) وقمـت بكتابـة اسـم الملـف مباشـرة فـسيبحث الـPHP عـن الملف في نفس المجلد الموجود بـه الـسكريبت ثم سيقوم بالبحـث عـن الملـف في المجلـدات الـتي تم تحديـدها في المـتغير use_include_path في ملـف php.ini

```
;;;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; UNIX: "/path1:/path2"
;include_path=".:/php/includes"
;
; Windows: "\path1;\path2"
;include_path=".;c:\php\includes"
```

مثال:

```
$fp=fopen ("./data.txt"، "r"، 1);
```

fclose

عنـدما تنتهـي مـن التعامـل مـع الملـف، تحتـاج إلى إغلاقـه لكـي يـتم حفـظ التعـديلات عليـه، إذا تم إحبـاط سـكريبتك لأي سـبب أو أن الـسكريبت انتهـي عملـه فـإن الـPHP يقوم بإغلاق جميع الملفات تلقائياً.

تقوم الدالة ()fclose بإغلاق الملف عندما تريد إغلاقه وهي تحتاج إلى معامل واحد فقط وهو مقبض الملف الذي تريد إغلاقه.

مثال:

```
Fclose ($fp) ;
```

قراءة وكتابة الملفات

لقد تعرفنا الآن كيفية فتح وإغلاق الملف، لنقم الآن بالتعرف علي كيفية قراءة و كتابة البيانات من الملف،

Fread

تقوم هذه الدالة بقراءة واستخراج البيانات الموجودة في الملفات ووضعها بمتغير وهي تأخذ معاملين المعامل الأول هو مقبض الملف والعدد الثاني هو عدد الحروف المراد قراءتها..

مثال:

```
$fp=fopen("data.txt"،"r");
$data=fread($fp،10);
```

وخذ باعتبارك نقطتين وهما:

١- إذا مثلاً قرأت عشر حروف من الملف وكان في الملف عشرين حرف وقمت بطلب الدالة fread مرة أخرى فسيتم قراءة العشر أحرف الثانية..

٢- إذا كان في الملف أقل من عشر أحرف فسيتم قراءة الموجود.

Fwrite

تقوم هذه الدالة بالكتابة إلى الملف وتحتاج إلى عاملين وهي مقبض الملف والقيمة المراد كتابتها إلى الملف، فعلى افتراض أنك قد فتحت الملف والمقبض هو $fp فإننا نكتب الكلمة PHP إلى الملف بالطريقة التالية:

```
Fwrite ($fp،"PHP");
```

وهناك معامل ثالث لهذه الدالة يحدد كم حرفاً سنقوم بكتابته من القيمة الحرفية الموجودة في المعامل الثاني فلو كتبنا مثلاً

```
Fwrite ($fp،"PHP"، 1);
```

فسوف يتم كتابه أول حرف فقط...

قراءة وكتابة الحروف في الملفات

Fgetc

تستخدم هذه الدالة لقراءة حرف واحد من الملف في كل مرة، وهي تستخدم معاملاً واحداً وهو مقبض الملف

وتقوم بإرجاع حرف واحد من الملف أو (False) عند الوصول إلى نهاية الملف..

Feof

تقوم هذه الدالة بخدمتنا في هدف بسيط وشيء ممتاز وهي معرفة إذا ما كنا قد وصلنا إلى نهاية الملف عند قراءته وتقوم بإرجاع (true) عند الوصول إلى نهاية الملف أو حصول خطأ ما، وهي تأخذ معاملاً واحداً وهو مقبض الملف.

فقد تكون مثلاً تريد أن تتأكد أن المؤشر لم يصل إلى نهاية الملف بعد استخدامك لأحد الدوال التي تقوم بنقل المؤشر من مكان إلى آخر، عند ذلك ستكون هذه الدالة مفيدة لتخبرك إذا ما وصلت إلى نهاية الملف أو لا...

تطبيق عملي:

قم بإنشاء ملف سمه file.txt واكتب فيه أكثر من سطر ثم قم بإنشاء ملف PHP وسمه بأي اسم وضع فيه الشيفرة التالية ثم اختبره، لكي ترى عمل الدالتين:

```
<?
$fp= fopen("file.txt"، "r");
While (!feof($fp))
{
 $char=fgetc($fp);
    echo $char;
} ?>
```

Fgets

إذا استخدمنا الدالة fgetc لقراءة الملفات الطويلة فإنها ستأخذ وقتاً وعمراً حتى تتم قراءتها ، يقوم الـPHP بتوفير دالة fgets لتساعدنا في قراءة عدد محدد من البايتات وهي تأخذ معاملين، المعامل الأول هو مقبض الملف والمعامل الثاني هو عدد الحروف المراد قراءتها +١ ، فإذا مثلاً أردت قراءة ملف يتكون من خمس حروف فسيكون المعامل الثاني للدالة هو الرقم ٦ وتتوقف الدالة عند حدوث أحد من ثلاث حالات

الأول: هو إذا تم قراءة عدد البايتات المحددة.

الثاني: إذا تم الوصول إلى نهاية سطر في الملف.

الثالث: إذا وصلت إلى نهاية الملف.

مثال:

```
 "r");،$fd = fopen ("/tmp/inputfile.txt"
while (!feof ($fd)) {
 4096);،     $buffer = fgets($fd
    echo $buffer;
}
fclose ($fd);
```

Fputs

تقوم بنفس وظيفة الداله fwrite وتأخذ نفس معاملاتها ونفس طريقتها..

القراءة داخل الملفات

File

تحتاج هذه الدالة إلى معامل واحد هو مسار الملف ولا تحتاج إلى مقبض، وعملها هو قراءة ما بداخل الملف وتخزينه سطراً سطراً في مصفوفة حيث إن هذه المصفوفة تقوم بأخذ كل سطر في الملف كأنه عنصر لوحده وتظل السطور سطوراً (أي أن المصفوفة تحتفظ بالمعامل للسطر الجديد (\n) بداخلها)، هذه الدالة لا تحتاج إلى

مقبض للملف بل تحتاج إلى مسار الملف فقط، وهـي تقـوم بفتح وقـراءة وإغـلاق الملـف تلقائياً...

وكغيرها من الدوال فإنها تستطيع قراءة صفحات الإنترنت الخارجية..

مع ذلك يستحسن أن لا تقوم باستعمال هـذه الدالـة لقـراءة الملفات الطويلة لأنهـا تقـوم باستخدام قدر كبير من الذاكرة المحجوزة للـ PHP وقد تستخدمها كلها...

مثال:

```
<?
$fcontents = file ('file.txt');
 $line) = each ($fcontents)) {،while (list ($line_num
echo "<b>Line $line_num:</b>  $line  <br>\n";
}
?>
```

Fpassthru

تقوم هـذه الدالـة بقـراءة محتويـات الملـف بدايـة مـن النقطـة الـتي توقـف منهـا المؤشـر الوهمي عند أي عمليـة قـراءة أخـرى، وتقـوم بـالتوقف عنـد نهايـة الملـف وتقـوم بـإغلاق الملـف مـن تلقـاء نفـسها لـذلك لا داعـي لإغـلاق الملـف بواسـطة الدالـة fclose بعـد استـخدامك لهذه الدالـة، وتقـوم الدالـة بقـراءة المحتويات وطباعتهـا بـشكل قياسـي، وهي تحتاج إلى معامل واحد فقط وهو مقبض الملف...

مثال:

```
<?
"r");،$fp=fopen("file.txt"
fpassthru($fp)
?>
```

Readfile

تقوم هذه الدالة بقراءة جميع محتويات الملف ولا تحتاج إلى مقبض بل إلى مسار الملف فقط وتقوم بقراءة كامل محتويات الملف ثم طباعتها بشكل قياسي وتقوم بإرجاع عدد البايتات التي تم قراءتها أو (false) عند حدوث خطأ ما

```
<?
Readfile ("file.txt");
?>
```

الوصول العشوائي إلى الملفات

أخبرناك سابقاً بأن هناك طريقة تجعلك تتحكم في التحكم بالمؤشر الوهمي للملف والوصول إلى أي مكان في الملف أو عند أي حرف تريده، بالدوال السابقة كنا عندما نصل إلى حرف معين مثلاً بدالة من الدوال فإننا نقوم بإغلاق الملف ثم نعاود فتحه كي نكمل القراءة من عند الحرف الذي تم الوصول إليه ولكن هذه الطريقة غير عملية نهائياً.....

يوصل لنا الـPHP بعض الدوال التي تمكننا من الوصول إلى الملف بالمكان الذي نريده ومن هذه الدوال:

Fseek

تحتاج هذه الدالة إلى عاملين، العامل الأول هو مقبض الملف $fp والعامل الثاني هو عبارة عن رقم صحيح يسمونه كمصطلح بالـ(offset) أي المكان الذي سيتوقف فيه المؤشر، سيقوم الـPHP بالتحرك في الملف إلى أن يصل إلى المكان الذي تم تحديده.. أي إنه إذا كان في الملف سطر واحد مكون من عشرة حروف وقمنا بجعل الـoffset خمسة، سيقوم الـPHP بالتحرك حتى يصل إلى نهاية الحرف الخامس...
وهناك معامل ثالث اختياري لهذه الدالة ويسمونه كمصطلح بالـ (whence) وله إحدى ثلاث خيارات:

Seek_set ويقوم بقراءة الملف من بدايته حتى يصل إلى المكان المطلوب بالـoffset

Seek_cur يقوم بالقراءة من المكان الحالي حتى يصل إلى المكان المطلوب بالـoffset

Seek_End يقوم بالقراءة من نهاية الملف حتى يصل إلى المكان المحدد بالoffset

تعتبر هذه الدالة نادرة في عملها (أو كما يسميها المبرمجون شاذة) بسبب أنها تقوم بإرجاع القيمة (٠) عند نجاحها والقيمة (-١) عند حصول خطأ ما..

مثال:

قم بفتح ملف واكتب فيه ثمان حروف متتالية ثم قم بحفظه باسم file.txt ثم قم بوضعه مع ملف PHP فيه الشيفرة التالية، ثم بعد ذلك شغل ملف الـPHP وانتظر النتيجة:

```
 <?
$fp = fopen("file.txt");
SEEK_SET);،4،fseek($fp
fpassthru($fp);
?>
```

Ftell

هذه الدالة من الدوال المفيدة فهي تقوم بإرجاع مكان الـoffset (أو المؤشر الوهمي) في الملف وتحتاج إلى معامل واحد وهو مقبض الملف...

```
<?
$fp = fopen ("file.txt");
$p = ftell($fp);
echo $p;
?>
```

Rewind

تقوم بإرجاع المؤشر إلى بداية الملف...

```
<?
$fp = fopen ("file.txt");
rewind($fp)
?>
```

جلب معلومات الملف

يوفر الـPHP دوال تساعدنا في معرفه حجم الملف وما إذا كان الملف موجوداً أم لا من هذه الدوال:

File_exists

تقوم هذه الدالة بالقيام بالتأكد ما إذا كان الملف موجوداً أم لا وهي تحتاج على معامل واحد وهو مسار الملف، وتقوم بإرجاع true (١) إذا كان الملف موجوداً و false إذا كان الملف غير موجود.

```
<?
$Th=File_exists("file.txt");
echo $Th ;
?>
```

Filesize

تقوم هذه الدالة بإرجاع حجم الملف بالبايتات أو false عند حصول خطأ...

دوال الملفات المتعلقة بالوقت:

هذه الدوال تقوم بإرجاع معلومات مفيدة عن وقت التغيير الذي طرأ على الملف أو آخر مره تمت قراءته وهي على حسب نظام التشغيل فإذا كان نظام السيرفير هو يونكس أو لينوكس ستقوم الدوال بإرجاع الوقت بنظام (timestamp) وهو الوقت مترجم إلى عدد الثواني منذ صدور يونكس ومولده على العالم، بينما تقوم بإرجاع وقت التعديل على نظام الويندوز مباشرة...

يقوم الـPHP بتزويدنا بدالتين لمعرفة الوقت:

Filectime وتقوم بإرجاع آخر وقت تم فيه التغيير على الملف على شكل timestamp ويشمل هذا آخر وقت تم فيه إنشاء الملف أو الكتابة إليه أو تغيير تراخيصه...

Filemtime تقوم بإرجاع آخر وقت تم فيه التعديل على الملف على شكل timestamp ويشمل هذا إنشاء الملف أو تغيير محتوياته...

تقوم الدالة getdate بعمل مفيد وهو تحويل الوقت من timestamp إلى الوقت العادي

الملكية والتراخيص

على أنظمة تشغيل اليونكس مثل يونكس ترتبط الملفات مع مستخدم خاص أو مجموعة من المستخدمين (group) وتحتوي على علامات وتراخيص تقوم بتوضيح من له صلاحية على استخدامها..

يمكننا أن نلخص التراخيص كالتالي:

١ / ممتلك الملف (owner)، بشكل افتراضي، وهو المستخدم الذي تم استخدام حسابه في استخدام الملف.

٢ / مجموعة من المستخدمين (group)، بشكل افتراضي، المجموعة التي يكون ضمنها مالك الملف.

٣ / جميع المستخدمين (all) كل شخص له حساب على النظام.

المستخدمين والمجموعات في أنظمة اليونكس يتم تعرفهم عن طريق رقم (ID) مثلما يتم تعريفهم عبر أسمائهم، إذا كنت تريد معرفة معلومات شخص عن طريق رقمه، يمكنك استخدام هذه الدالة:

Posix_getpwind

التي ستقوم بإعطائنا مصفوفة تحتوي على المعلومات التالية:

Name	اسم المستخدم الذي يدخل به في حسابه.
passwd	كلمة السر المشفرة للمستخدم.
uid	رقم الحساب للمستخدم.
gid	رقم حساب المجموعة التي فيها المستخدم.
gecos	اسم المستخدم الكامل، رقم هاتف ومعلومات إضافية.
dir	المجلد الرئيسي للمستخدم.
shell	المسار الرئيسي لحساب المستخدم.

Posix_getgrgid

تقوم هذه الدالة بإرجاع مصفوفة عن معلومات المجموعة، وهي تحتاج إلى معامل واحد فقط وهو رقم الـID للمجموعة...

وسوف تحتوي على العناصر التالية:

Name	اسم المجموعة
Gid	رقم المجموعة
members	عدد أعضاء المجموعة

وهنـاك أيضـاً خمـس دوال تـساعدنا في معرفـة معلومـات أكـثـر عـن الملفـات وتحتـاج فقط إلى مسار الملفات.

Fileowner

تقوم بإرجاع رقم المعرف (ID) لمالك الملف...

Filegroup

تقوم بإرجاع رقم المعرف (ID) لرقم المجموعة التي يعتبر مالك الملف ضمنهم..

Filetype

تقوم بإرجاع رقم نوع الملف وقد تعود بإحدى هـذه القيم (file، dir، char، fifo، link، block) والذي يهمنا منهم هو file وdir...

Is_dir

وتقوم بإرجاع True إذا كانت قيمة المسار هو مجلد..

Is_file

وتقوم بإرجاع True إذا كانت قيمة المسار هي ملف..

الحصول على اسم الملف من وسط مسار الملف..

Basename()

مثال:

```
<?
$path = "/home/httpd/html/index.php3";
$file = basename ($path);
echo '$file <br>';
".php3"); ،$file = basename ($path
echo '$file <br>';
?>
```

هذه الدالة مفيدة جداً للحصول على الملف من وسط مسار مجلد..

نسخ، اعادة تسمية وحذف الملفات

تسمح لك الـPHP بنسخ، وإعادة تسمية، وحذف والدوال التي تستخدم لتنفيذ هذه العمليات هي

Copy ()

تقوم بأخذ قيمتين حرفتين وتشير إلى مصدر الملف الرئيسي الذي يوجد فيه الملف والمصدر الهدف الذي سيتم نسخ الـPHP إليه...

```
<?
 $file.'.bak')) {،if (!copy($file
print ("failed to copy $file...<br>\n");
}
?>
```

Rename

نستطيع الآن استخدام هذه الدالة لإعادة تسمية الملف وهي تحتاج إلى قيمتين حرفتين وهي مصدر الملف أو مكانه واسمه الرئيسي ثم الاسم الجديد الذي تريد إعادة التسمية به..

مثال:

```
<?
'newfile.txt');،Rename ('file.txt'
?>
```

Unlink()

تحتاج إلى قيمه حرفية واحدة وهي مسار الملف الذي تريد حذفه.

```
<?
unlink ('file.txt');
?>
```

العمل مع المجلدات

مثلما تعاملنا مع الملفات في الـPHP فإننا نتعامل مع المجلدات، فهناك دوال للمجلدات تتطلب مقبض المجلد، وهناك دوال تحتاج فقط إلى القيمة الحرفية فقط وبدلاً من الإطالة دعنا نقوم بالدخول في الموضوع مباشرة.

Opendir تقوم بفتح المجلد وإعطائنا مقبض المجلد.

Closedir() تقوم بإغلاق المجلد المفتوح وتحتاج فقط إلى مقبض المجلد...

Readdir تقوم بقراءة المدخل الحالي للمجلد...

Rewindir تقوم بإرجاع المدخل إلى الصفر..

Chdir للانتقال إلى مجلد آخر، وتتطلب المسار للمجلد الذي تريد الانتقال إليه..

Rmdir تقوم بمسح مجلد، ولكن يجب أن يكون المجلد خالياً من أي ملفات أو مجلدات، وتتطلب مسار المجلد الذي تريد مسحه..

Mkdir تقوم بإنشاء مجلد جديد وتتطلب أن يكون هذا المجلد غير موجود مسبقاً وتحتاج إلى قيمتين وهما اسم المجلد الجديد مع مساره، والترخيص المطلوب له..

Dirname تقوم بإعطائنا اسم المجلد الحالي الذي فيه الملف، وتحتاج إلى مسار الملف..

تطبيق عملي:

أنشئ مجلد اسمه tmp في مجلد الـ htdocs وضع فيه ملفات، ثم أنشئ ملف اسمه test.php في مجلد الـhtdocs واكتب الشيفرة التالية ثم شغله:

```
<?php
if ($dir = @opendir("/tmp")) {
while($file = readdir($dir)) {
echo "$file\n";
}
closedir($dir);
}

?>
```

Dir()

عبارة عن كائن يحتوي على ثلاث وظائف.. ونقوم بإعطائه مسار المجلد الذي نريده أن يتعامل معه ثم بعد ذلك نقوم بوضع قيمته في متغير يقوم بوراثة صفاته.

خصائص الكائن:

handle تقوم بإعطائك مقبض المجلد..

Path تقوم بإعطائك المسار للمجلد..

Read تقوم بإعطائنا المجلدات اعتماداً على المؤشر الحالي للمجلد..

Rewind تقوم بإرجاع مؤشر المجلد من الصفر.. تقريباً نفس عملية rewinddir..

Closedir تقوم بإغلاق المجلد..

دعنا نأخذ فكرة عن طرف التخزين في البداية وكيف كانت على الإنترنت في السابق....

كان من أكثر طرق التخزين انتشاراً في السابق على الإنترنت وربما هو لا يزال يستخدم في بعض المواقع والمنتديات يعتمد على الملفات.... فكان صاحب الموقع الذي

لديه هذه الطريقة في تخزين البيانات خوف وتعب من فقدانها مثلاً وكان الشبح الذي يكدر عليه صفوة نجاح موقعه هو عمل نسخ احتياطية لهذه الملفات لكي يتمكن من استرجعها في حال فقدانها فكانت هذه العملية تأخذ وقتاً وجهداً ومالاً.... كما كان من عيوب تخزين البيانات في الجداول ضغط الخادم أو (server) في حال الاستعلام عن معلومة معينة والبحث عنها كما أنه يستهلك الكثير من ذاكرة هذا الخادم في عملية بحث معينة فهو يحجز مساحة ليست بالهينة في عملية بحث عن اسم مثلاً أو ما شابهها.

ربما يتردد عند البعض ذالك السؤال وهو....ما هي قواعد البيانات بالضبط ؟

قواعد البيانات ببساطة جمع المعطيات أو المدخلات.
كل قاعدة بيانات ربما تتكون من جدول (Table) واحد أو عدة جداول هذه الجداول تحتوي علي أعمدة وصفوف تهيكل البيانات وترتبها.... سوف أجعل لك مهمة اكتشاف فوائد قواعد البيانات في آخر الفصل.
لترى الجدول الذي بالأسفل كمثال:

#Table "Coustomers"

Id	Fname	Lname
025	عبد الواهب	صالح
044	محمد	خالد

كما تلاحظ الجدول قسَّم البيانات إلى صفوف...مع كل إضافة عميل جديد سوف يكون هناك صف (سجل) جديد... ربما لو تطلق لخيالك العنان سوف تلاحظ أن هذا الجدول مشابه للدولاب والصفوف رفوف فإذا أردت أن تضيف كتب أو ملابس

أو أي كان سوف تضيفها في رف جديد.. كما يحصل في إضافة عميل جديد سوف تضيفه في صف (سجل) جديد.

البيانات في كل صف قسمت إلى مدى أبعد في الخلايا (أو الحقول)، كل من هذه البيانات تحتوي على قيمة محددة وصفة محددة. على سبيل المثال محمد خالد سوف ترى أن هذا العميل انقسمت بياناته في الحقل إلى id والاسم الأول والاسم الأخير.

الصفوف في الجدول ليس لها ترتيب معين.. يمكن أن يكون الترتيب أبجدياً ويمكن أن يكون باسم العضو أو باسمه الأخير أو بأي معيار آخر يمكن أن تحدده مسبقاً لترتيب الصفوف ولهذا يكون من الضروري تحديد طريقة ليسهل عليك تحديد صف (سجل) معين....في المثال السابق نستطيع إخراج السجل من بين باقي السجلات بـ id وهو رقم العميل الذي هو عدد فريد لا يتكرر في أي صف (سجل) أخر وسبب استنادي في استخراج السجل علي id لأنه ربما يكون هناك عميلان لهما نفس الاسم.... وهذا ليس شرط أن يكون للجدول مفتاح فريد لكن هنا حددناه لكي يتم استخراج السجلات المطلوبة بسهولة وبسرعة أكبر.

العلاقات

الكثير من قواعد البيانات اليوم هي نظم إدارة قواعد بيانات علائقية (relational database management systems) تختصر في RDBMS، قواعد البيانات العلائقية هذه عبارة عن مجموعة من الجداول أو نموذج من الجداول النموذجية المتعددة التي تحتوي على معلومات مترابطة. ربما تسمع أيضاً الكثير عن SQL وهي اختصار ل(Structured Query Language) وهي تسمح لك أن توحد هذه المعلومات من الجداول المترابطة وبذالك تسمح لك بإنشاء وتحليل العلاقات الجديدة.

المثال السابق للعملاء كان عبارة عن جدول واحد فقط، ولذلك لن تحتاج إلى ربط بينه وبين جدول آخر لأنه لا يجدي.

لكن إذا كان هناك أكثر من جدول وكانت هذه الجداول مترابطة مع بعضها البعض في البيانات سوف تلاحظ أنك بحاجة إلى نظم إدارة البيانات العلائقية (RDBMS).... فلنرى هذه المثال لكي تتضح الصورة أكثر:

"Table "Coustomers#

Id	Fname	Lname
025	عبد الواهب	صالح
044	محمد	خالد
022	حمد	طارق

"Table "Address#

Id	Tel	Street	City	Country
044	018522	شارع الأهرام	القاهرة	مصر
022	01225505	مساكن برزة	دمشق	سوريا
025	0122505	طريق الأربعين	الكويت	الكويت

" Table "Account#

Id	accountb
044	10.0000
025	20.0000
022	20.000

كل من هذه الجداول الثلاثة كيان مستقل.... لكن تلاحظ أنهم مرتبطين مع بعضهم البعض ب(id)، على سبيل المثال بإمكاننا أن نعرف رصيد العميل عبد الواهب صالح من id، كما يمكننا معرفة اين يسكن حمد طارق وكم رقم الهاتف وأيضاً يمكننا أن نعرف من هو صاحب الرصيد ٢٠,٠٠٠ أيضاً كم واحد من مدينة القاهرة والكثير الكثير..... ربما اتضحت لك أهمية العلاقات.

إذا عرفنا أن العلاقات هي الأساس الجوهري لنظم قاعدة البيانات العلائقية، بجعلها مرنة وسهلة بحث تتمكن من ربط السجلات المختلفة مع بعضها البعض في الجداول.

المفتاح الأجنبي

سوف تلاحظ أن حقل (id) الذي يظهر في الجداول في الثلاثة والذي جعل من الممكن ربط الجداول المختلفة معاً أنه مفتاح أجنبي لأنه بالأصل مفتاح فريد (primary key) في جدول (COUSTOMERS)...

ليس ضرورياً أن يكون هناك مفتاح أجنبي في كل جدول ولكن يتم إضافة على حسب حاجتك فإذا كنت تريد ربط بيانات الجداول مع بعضها فسوف تحتاج إليها.

في كل جدول يوجد به المفتاح الأجنبي سوف يكون له مرجعية للجدول الأصل فمثلاً هنا المرجعية ستكون جدول (customers).... بمعنى أن المفتاح الأجنبي سوف يقوم بربط البيانات ما بين الجدول الأصل وبين الجدول الذي يتواجد به كمفتاح أجنبي....من هنا يتضح لنا مفهوم الاستقامة المرجعية وهذا مفهوم أساسي ومهم عندما تصمم قاعدة بيانات بأكثر من جدول سوف يكون للمفتاح الأجنبي قيمة ثابتة في جميع الجداول بمعنى لو كان قيمة المفتاح الأجنبي في جدوله الأصل عدد صحيح فسوف يكون بنفس القيمة في جميع الجداول.... ونقطة أخرى إذا حدث تحديث أو

تغيير أو حذف لأحد القيم في المفتاح الأجنبي فسوف تتم في جميع الجداول...هذا هو مفهوم الاستقامة المرجعية.

كثير من قواعد البيانات اليوم يتم تعديل القيم بها تلقائياً كميكروسوفت أكسس وبعض قواعد البيانات الأخرى، لكن هناك بعض قواعد البيانات التي تحتاج إلى تعديل يدوي على كل قيمة يتم التعديل عليها... وهذا لاشك أنه متعب!!

الفهرسة

لو كان لديك جدول به الكثير من السجلات، يمكنك أن تستعلم بسرعة كبيرة عن أياً من هذه السجلات بواسطة "فهرسة" كل السجلات. هذا المفهوم تقريباً شبيه جداً بالفهرس الذي يوجد نهاية كل كتاب...كما يسهل عليك هذا الفهرس الموجود في الكتاب في سرعة البحث عن المواضيع التي يتضمنها الكتب، نفس الكلام ينطبق على فهرسة السجلات في الجدول....دعنا نرى مثالاً لتتضح الصورة:

```
= 220; SELECT * FROM names WHERE ID
```

سوف يقوم هذا الاستعلام في البحث في جميع المعلومات وإرجاع قيمتها بشرط أن يكون رقم السجل (الفهرس) ٠٢٢

هنا سهلت علينا المهمة كثيراً وذلك لأن السجلات مرتبة بأرقام فلكل سجل رقم فريد يميزه عن الآخر وبهذه الحالة سوف يقوم هذا الاستعلام السابق بجلب جميع بيانات العميل "حمد طارق "....

تدريب عملي

سوف نبدأ بأخذ فكرة عن البرنامج الذي سوف أشرح عنه وهو عبارة عن دليل مواقع بسيط جداً يعتمد علي قواعد البيانات.
بالطبع لا يوجد فيه تقسيمات والسبب لأننا سوف نضطر إلى وضع أو خلق أكثر من جدول (Table) في قواعد البيانات وسوف نضطر إلى وضع علاقات بينهم.

وهذا حالياً يعقد الموضوع فسأكتفي الآن بجدول واحد (one table) وسوف أشرح إذا اتسع لدي الوقت الكثير عن (sql) عامة وعن العلاقات (Relationstips) لأنها مثيرة أيضاً جداً.

نبدأ بالخطوة الأولى وهي قاعدة البيانات الخاصة بدليل الموقع الذي لدينا ، فالبيانات والمعلومات التي غالباً ما يهتم بها دليل الموقع هي عنوان الموقع واسم الموقع والبريد الخاص بالموقع.

هذا هو الكود الخاص **بإنشاء قاعدة البيانات الخاصة بدليل الموقع.**

```
CREATE TABLE dalal (
، id int not null auto_increment
،sitename varchar(100)
،add_date date
،email varchar(100)
،site_url varchar(100)
 ،description text
primary key (id));
```

هنا سوف يخزن في قواعد البيانات المعلومات التالية:

id وهي المرشح الأساسي والفريد الذي يُتعرف على الجدول بواسطته.

sitename وهو اسم الموقع وهو من نوع char

add_date تاريخ إضافة الموقع وهو من نوع date

email البريد الإلكتروني وهو من نوع char وطوله نفس طول اسم الموقع (١٠٠)

site_url وهو عنوان الموقع أيضاً من نوع char

سوف أكتفي بشرح id لأنه ربما يكون جديد على البعض وسوف أشرح عمله وليس المهم أن تفهم عمله في هذه الخطوة إنما سوف تفهمها بالتفصيل في الخطوات القادمة بعد قليل إن شاء الله.

id هو عبارة عن المرشح الأساسي لهذا الجدول (primary key)

وتلخص ‌في آخر سطر من الجدول وجود:

```
primary key (id)
```

ربما ترجمة المصطلحات العلمية وخاصةً ‌في الكمبيوتر تضر أكثر مما تنفع وهذا سبب وجيه لي ‌في كتابتي السكريبت مهما كان بلغة إنجليزية ومن ثم تعريبها.

المهم نحن الآن نريد أن نعرف ماذا يعني id سوف أعطي مثالاً لكي يتضح فلو أردنا مرشحاً أساسياً أو كوداً فريداً نميز به هذا الجدول فلن نستطيع أن نضع هذا الكود الفريد لاسم الموقع لأنه ربما يكون لدي موقع بنفس الاسم والاختلاف بينهما ‌في الوصلة وربما يكون هناك موقع بعنوانين مثل سوالف وياهوو إذا اسم الموقع لا يمكن أن نضعه كوداً أساسياً وأيضاً نفس الكلام ينطبق على عنوان الموقع وأيضاً على التاريخ وحتى البريد لا نستطيع أن نضع البريد هو المرشح الوحيد والأساسي للجدول ربما يأتي هذا لكن منطقياً لا يجوز.

ربما ترى id من نوع عدد صحيح (int) لكن ربما يصعب عليك الذي بجانبه وهو (not null) وهي تعني بعدم السماح لهذا الحقل أن يكون بدون قيمة.

not null تعني بكل وضوح ربما يسمح بقيمة فارغة ربما تفكر ‌في الصفر فهذا غير صحيح.

وترى أيضاً هذه العبارة بجانب not null وهي (auto_increment) وهي تعني بكل وضوح الإضافة التلقائية أو الأتوماتيكية ودائماً ما يبدأ بالقيمة (١)

وهذا يعني مع كل إضافة موقع سوف تزيد قيمة (id) وهي تبدأ من (١)

config.php

وهو عبارة عن ملف التوجيه الذي يحتوي معلومات الموقع وهي المعلومات التالية:

```
<?
//this moudel config
$dbname = "links";
$dbuname = "root";
$dbpws = "";
$dbhost = "localhost";
?>
```

لا يحتاج إلى شرح لأنه واضح فالذي جرب أن يركب أي برنامج php يعتمد على قواعد البيانات فالأكيد سيعدل في ملف مشابه له أو نفسه ومع ذلك سنشرحه:

الآن نبدأ في **add.php** وهذا هو:

```
<html>
<head>
<style type="text/css">
verdana; font-size:10pt}،body {font-family:verdana
verdana; font-size:10pt}،TD {font-family:verdana
#header {color:white; font-weight:bold; font-
verdana}،family:verdana
A {color:navy; text-decoration:underline}
A:hover {color:red}
A:visit {color:navy}
</style><title>Site Insertion</title></head>
<?
if ($action == 'addsite')
{
include("config.php");
 $dbpw);، $dbuname،     mysql_pconnect($dbhost
@mysql_select_db("$dbname") or die ("Unable to select database"
);
،'$email'،''،'$name'،     mysql_query("insert into dalal values (''
'$des')");،'$siteurl'
mysql_close();
echo "Thank you for add site ";
}
?>
<body bgcolor=white>
<form method="POST" action="add.php?action=addsite">
<p>SiteName <input type="text" name="name" size="40" style="
border-style: double; border-color: #000080"></p>
<p>Email       <input type=
"text" name="email" size="40" style="border-
style: double; border-color: #000080"></p>
<p>SiteUrl    <input type="text" name="siteurl
```

```
" size="40" style="border-style: double; border-
color: #000080" value="http://"></p>
<p>description<textarea rows="5" name="des" cols="33" style="
border-style: double; border-color: #000080"></textarea></p>
<p align="center"><input type="submit" value="Submit" name="
B1" style="border-style: double; border-
color: #000080"><input type="reset" value="Reset" name="B2" s
tyle="border-style: double; border-color: #000080"></p>
</form>
<br>
<a href="show.php"><b>View</b></a>
<br>
</body>
</html>
```

من رؤيتك له تلاحظ أنه عبارة عن html و php لكن كود php أكثر

بالنسبة html فهي بسيطة بإمكانك فهمها وهي عبارة عن حقول تطلب من المستخدم إدخال معلومات الموقع سوف أبدأ في شرح كود php

فتلاحظ بدايته ونهايته بهذين الوسمين ?> و ?<

وبعدها يأتي شرط التحقق وهو:

```
if ($action == 'addsite')
```

فهذا يعني التحقق من إضافة جديدة على الحقول الخاصة بعملومات الموقع وتلاحظ أن التحقق من هذا الشرط بهذه القيمة ('addsite') بين وسمين اقتباس وحيد ربما يكون هناك بعض التساؤل قليلاً عليه لكن لو تلاحظ في كود html في form وضعت الانتقال بعد إضافة المعلومات إلى (add.php?addsite) هنا

```
<form method="POST" action="add.php?action=addsite">
<p>SiteName <input type="text" name="name" size="40" style="
border-style: double; border-color: #000080"></p>
<p>Email       <input type=
"text" name="email" size="40" style="border-
style: double; border-color: #000080"></p>
<p>SiteUrl    <input type="text" name="siteurl
```

```
" size="40" style="border-style: double; border-
color: #000080" value="http://"></p>
<p>description<textarea rows="5" name="des" cols="33" style="
border-style: double; border-color: #000080"></textarea></p>
<p align="center"><input type="submit" value="Submit" name="
B1" style="border-style: double; border-
color: #000080"><input type="reset" value="Reset" name="B2" s
tyle="border-style: double; border-color: #000080"></p>
</form>
```

هذا هو كود الفورم الخاص بإدخال معلومات الموقع فإذا لاحظته في بدايته هنا

```
<form method="POST" action="add.php?action=addsite">
```

وضعت الإرسال بالـ (post) وقمت بوضع قيمة (action) الصفحة (add.php?addsite)

هذا يعني أن قيمة (action) في حال ضغط submit ستكون ما هو مكتوب بعد علامة الاستفهام (؟) بعد اسم الصفحة وهي (add.php?addsite)

أتمنى أنك فهمت المقصود.

بعد ذلك إذا تحقق الشرط ووجد من أرسل سوف ينفذ الأوامر والأولى منها.

```
include("config.php");
```

وأمر include يعني استجلاب هذا الملف وقيم المتغيرات الموجودة به

بعد هذا الأمر طلبت الاتصال بقواعد البيانات لكي يتم تجهيز الجدول للإضافة وهو هنا:

```
mysql_pconnect($dbhost، $dbuname، $dbpw);
@mysql_select_db("$dbname") or die ("Unable to select database"
);
```

الدالة mysql_pconnect تطلب الاتصال بخادم قاعدة البيانات على أساس المعلومات الموجودة بها

وهنا نحن استعنا بعد الله بالمتغيرات الموجودة في ملف config.php

وربما هذا يوضح لنا أكثر فائدة الدالة include

ربما لاحظت في طلب الاتصال في الدالة mysql_pconccet في السطر الثاني بعدها هذا

```
@mysql_select_db("$dbname") or die ("Unable to select database");
```

هنا في الدالة mysql_select_db وضعنا قيمة المتغير dbname وهو اسم قاعدة البيانات

فهذه الدالة تقوم بتحديد اسم قاعدة البيانات المراد الاتصال بها وهذا ربما يوضح استخدام علامة @

لكي يعمل كل شيء بنفس الوقت

بعد هذا نلاحظ السطر التالي:

```
mysql_query("insert into dalal values ('','$name','','$email','$siteurl',
'$des')");
```

قمنا في هذا السطر بتنفيذ الدالة mysql_query وهذه الدالة تقوم باختصار تنفيذ استعلامات (sql) وفي هذا السطر استخدمنا أمر الاستعلام المراد تنفيذه بواسطة الدالة mysql_query هو (Insert)

هذا الأمر أو الدالة في sql تقوم بإضافة البيانات في قاعدة البيانات والصيغة أو القاعدة الثابتة له هي:

```
INSERT INTO tablename VALUES (var and values)
```

INSERT INTO

هذا ثابت لدينا بعده يأتي اسم الجدول الذي نريد إضافة البيانات بداخله

وبعد ذلك تأتي كلمة VALUES (var وتفتح قوس وتيحط المتغيرات والقيم المراد إضافتها بأقواس اقتباس (" ") وتنهيه بفاصلة منقوطة ;

تلاحظ حين استخدمت mysql_query و insert لم أضع القيم بل وضعت المتغيرات التي تحمل القيم المراد تخزينة في جدول dalal ربما هذا يسهل علينا الكثير

بعد ذلك تأتي الدالة mysql_colse وهي تعني طلب إغلاق الاتصال بخادم قاعدة البيانات

بهذا انتهينا من ملف add.php و config.php

هنا سوف نعرف كيف نستخرج البيانات الموجودة داخل قاعدة البيانات
وهي في غاية السهولة

بدايةً سوف أضع كود **show.php** كاملاً واشرحه مثل سابقه:

```
<html>
<head>
<style type="text/css">
verdana; font-size:10pt}،body {font-family:verdana
verdana; font-size:10pt}،TD {font-family:verdana
#header {color:black; font-weight:bold; font-
verdana}،family:verdana
A {color:navy; text-decoration:underline}
A:hover {color:red}
A:visit {color:navy}
</style>
</head><titel><center><b>view The sites</b></center><br><
center><a href='add.php'>Add Your Site</a></center> <hr color
="#000080"></titel>

<body>
<?php
include("config.php");
 $dbpw);، $dbuname،    mysql_pconnect($dbhost
@mysql_select_db("$dbname") or die ("Unable to select database"
);

$query="select * from dalal ";

$result=mysql_query($query);

mysql_close();

/*Display Results*/
```

```
$num=mysql_numrows($result);

$i=0;
while ($i < $num) {


"sitename");،$i،    $sitename=mysql_result($result
"add_date");،$i،    $add_date=mysql_result($result
"email");،$i،    $email=mysql_result($result
"site_url");،$i،    $siteurl=mysql_result($result
"description");،$i،    $description=mysql_result($result
"id");    ،$i،    $id=mysql_result($result
?>
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><b>SiteName:</b><a href='<?echo $siteurl;
?>'><i><? echo $sitename;?></i></a>
<p><b>Date Add:</b>  <i> <?echo $add_date;?></i></p>
<p><b>description:</b><?echo$description;?></p>
<p><b>URL:</b><i><?echo$siteurl;?></i></p>
<p><b>Email:</b><i> <a href="mailto:<?echo$email;?>"><?ec
ho $email;?></a></i> </p>
</td>
</tr>
</table>
<a href='edit.php?id=<?echo $id;?>'><b><b>[Edit This Site]</b
></b></a>
<hr color="#000080">
<?
++$i;
}

?>
</body>
</html>
```

كما تلاحظ تقريباً شبيه بالملف add.php في عملية الاتصال لكن الجديد لدينا هنا

السطر الذي يلي عملية الاتصال وهو:

```
$query="select * from dalal ";
```

في هذا السطر قمنا بتعريف متغير يدعى query وقمنا بوضع قيمة هذا المتغير عملية الاستعلام عن المعلومات الموجودة في الجدول dalal

أظن هذا واضح لكن دعني أشرح عمل select فهي تقوم بتحديد الحقول بداخل الجداول أو الجدول وهي دائماً تستخدم في إخراج البيانات من الجداول فتلاحظ وضعنا قيمة الحقل أو الحقول في استعلامنا هذا علامة الضرب (*) وهي تعني تحديد جميع الحقول الموجودة وبعدها استعملنا from وهذا يعني من ودائماً يأتي بعده أسماء الجدول أو اسم الجدول الذي قمت بتحديد الحقول بداخله.

ببساطة واختصاراً لهذا السطر select * from dalal

حدد جميع الحقول من جدول الدليل.

هنا سوف يحدد جميع الحقول الموجودة في جدول الدليل ويقوم بتخزينها في المتغير query

بسيطة.

لا تنس أن تضعها بين أقواس اقتباس مزدوجة (" ")

بعد ذلك يأتي هذا السطر

```
$result=mysql_query($query);
```

قمنا هنا أيضاً بتعريف المتغير result وقمنا بتخزين قيمة الدالة mysql_qurey وقيمة المتغير الذي استعلمت به الدالة mysql_query في المتغير result

بوضوح هنا قمنا بالاستعانة بالدالة mysql_query ونسبنا إليها قيمة المتغير query الموجود به مسبقاً قيمة الاستعلام مسبقاً في السطر الذي يسبق هذا السطر وقمنا بهذا لكي يتم فرز جميع القيم الموجودة في الجدول dalal بواسطة الدالة mysql_query

بعد ذلك قمنا بإغلاق قاعدة البيانات بالدالة mysql_colse

وبعد ذلك قمنا بتنويه بسيط وتعليق يوضح أننا سوف نعرض ونطبع القيم الموجودة في الجدول في هذا السطر.

/❖Display Results❖/

بعد هذا السطر قمنا بتعريف متغير يدعى num وأضفنا القيمة إليه باستخدام الدالة mysql_numrows مسندين إليه قيمة المتغير result

الدالة mysql_numrows تقوم باسترجاع ومعالجة نتيجتنا المخزنة في المتغير result وتقوم بتخزينها في المتغير num على شكل مصفوفة

بعد ذلك قمنا بتعريف عداد يدعى i ووضعنا قيمته صفر لكي نستخدمه مع أمر التكرار while وقمنا بوضع الشرط في أمر التكرار while

i < num

هذا يعني إذا كان قيمة العداد i أصغر من قيمة المتغير num المتوجد بداخله جميع المعلومات فاستمر بطباعة التالي.

وهي هذه السطور.

```
$sitename=mysql_result($result،$i،"sitename");
$add_date=mysql_result($result،$i،"add_date");
$email=mysql_result($result،$i،"email");
$siteurl=mysql_result($result،$i،"site_url");
$description=mysql_result($result،$i،"description");
$id=mysql_result($result،$i،"id");
?>
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="100%"><b>SiteName:</b><a href='<?echo $siteurl;?>'>
<i><? echo $sitename;?></i></a>
<p><b>Date Add:</b>  <i> <?echo $add_date;?></i></p>
<p><b>description:</b><?echo$description;?></p>
<p><b>URL:</b><i><?echo$siteurl;?></i></p>
<p><b>Email:</b><i> <a href="mailto:<?echo$email;?>"><?echo $e
mail;?></a></i> </p>
</td>
</tr>
```

```
</table>
<a href='edit.php?id=<?echo $id;?>'><b><b>[Edit This Site]</b></b
></a>
<hr color="#000080">
<?
++$i;
}
```

هنا قمنا بالكثير في أمر التكرار WHILE فسوف يطبع جميع القيم الموجودة في الجدول كما رمزت إليها في هذا السطر

```
$sitename=mysql_result($result،$i،"sitename");
```

قمنا في هذا السطر والسطور الخمسة التي تليه بتعريف العديد من المتغيرات وقمنا بتخزين قيم المعلومات بداخله باستخدام الدالة mysql_result مسندين إليه المتغيرات result و i و الحقل الذي نريد إخراج القيمة منه
واستخدمنا في هذا السطر الحقل sitename

كما استخدمنا كود html لكي ننسق ونرتب شكل ظهور المعلومات
وفي آخر سطر قبل أن ننهي عملية التكرار while بالقوس } جعلنا قيمة العداد i تزيد بمقدار واحد مع كل لفة أو بمعنى طباعة قيم من قاعدة البيانات لكي يتم توقفها أرجو أن تنتبه لهذه الأمور الصغيرة لأنها من مشاكل البرمجة المشهورة.

هنا انتهينا من ملف show.php يتبقى لنا ملف edit.php وهو الملف الخاص بتعديل الموقع فربما يخطئ أحدهم في إدخال اسم أو عنوان موقعه فهنا يمكنك تعديلها حتى بعد تخزينها في قاعدة البيانات.
هذا آخر ملف سوف أشرحه اليوم من ضمن مشروع دليل المواقع وربما أشرح عملية الحذف والبحث في وقت آخر.

سوف أضع كود **edit.php** كاملاً كسابقيه

```
<html>
<head>
<style type="text/css">
body {font-family:verdana،verdana; font-size:10pt}
TD {font-family:verdana،verdana; font-size:10pt}
TH {font-size:10pt; color:white; font-weight:bold; font-
family:verdana،verdana}
A {color:navy; text-decoration:underline}
A:hover {color:red}
A:visit {color:navy}
#mySelect {font-family:verdana،arial; font-size:9pt}
</style>
</head>
<titel><center><b>Edit Site</b></center><br><center><a href='add
.php'><b>Add Your Site </b></a> |
<a href='show.php'><b>View Edit database</b></center></a> <hr c
olor="#000080"></titel>
<body>

<table border=0 cellpadding=4 cellspacing=0 width=600>
<? echo "<form method='post' action='$PHP_SELF'>"; ?>
<table border="1" width="100%" bordercolor="#000080" cellspacing="
1" cellpadding>
<tr>
<td width="100%">
<p align="center">Edit The Info </td>
<?
include("config.php");
mysql_pconnect($dbhost، $dbuname، $dbpw);
@mysql_select_db("$dbname") or die ("Unable to select database");

$query1 = "select * from dalal where id = '$id'";

$result1 = mysql_db_query('links'،$query1);

while($row = mysql_fetch_object($result1)) {
echo "<td align=middle>SiteNmae<br><input type=text name='name'
value='$row->sitename'></td>";
echo "<td align=middle>Add Date<br><input type=text name='add_da
```

```
te' value='$row->add_date'></td>";
echo "<td align=middle>Email<br><input type=text name='email' value
='$row->email'></td>";
echo "<td align=middle>SiteUrl<br><input type=text name='siteurl' val
ue='$row->site_url'></td></tr>";
echo "<td align=middle>Description<br><input type=text name='desc'
value='$row->description'></td></tr>";
$id = $row->id;
}
echo "</table><p>";
echo "<br><input type=hidden name='id' value='$id'>";
?>

<input type=submit name="update" value="Edit The Info">
</form>
</tr>
</table>
<?
if ($update) {
if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $
desc == "") {
die("<b>You left one or more fields blank.</b>");
}

$query2 = "update dalal set sitename='$name'، add_date='$add_date'،
email='$email'، site_url='$siteurl'، description='$desc'
where id='$id'";
mysql_db_query('links'،$query2);
echo "<b>Your record has been updated</b></p>";
}
mysql_close();
?>
</body>
</html>
```

هنا سوف يسهل علي شرح هذا الملف أكثر من سابقيه لأنه يكرر ما يفعله إخوانه لكن مع بعض التجديدات التي لا تذكر فالجديد هنا وفي هذا السطر

```
$query1 = "select * from dalal where id = '$id'";
```

هنا تلاحظ عرفنا متغير يدعى query1 وقمنا بتخزين ناتج الاستعلام بداخله فهذا ليس بجديد الجديد ما يوجد في الاستعلام فقد استخدمنا where وهي تعني

بالترجمة إلى العربية "بحيث" وتعني في البرمجة وخاصةً مع أوامر sql "بشرط" أي هنا اشترطنا في استعلامنا أن تكون قيمة المتغير id مساوية لنفس القيمة الموجودة في الجدول وهي id وربما هنا يسعني أن أوضح أكثر ما هو id ولماذا عرفته وجعلته primary key

هنا إذا ضغط أحد الذين أضافوا مواقعهم بعد أن أخطأ في إضافة موقعه على edit فسوف يعتمد على الرقم المخزن في id وهو كما سبق وأن عرفتها في قاعدة البيانات عدد صحيح ويزداد تلقائياً فلو كان لدينا مثلاً في قاعدة البيانات أربعة مواقع فلن يجري التعرف على الأربعة مواقع ومعلوماتهم إلا عن طريق id لأن لكل موقع منهم رقم، بمعنى الأول منهم ١ والأخير ٤ أرجو أنك فهمته وأرجو أن تحاول تشغيل السكريبت لكي تفهمه أكثر وأكثر.

بعد ذلك يأتي هذا السطر

```
$query1);،$result1 = mysql_db_query('links'
```

هنا أيضاً عرفنا المتغير result وخزنا بداخله قيمة استعلام الدالة mysql_db_query وأسندنا إليها اسم قاعدة البيانات والمتغير query

الدالة mysql_db_query وهي مشابة في عملها مع mysql_query ما عدا الفرق بينهم أنك تستطيع تحديد اسم قاعدة البيانات الذي تريد الاستعلام عنه واستعملتها هنا لكي يتم إعطاؤنا النتائج بوضوح أكثر.

بعد هذا السطر يأتي أمر التكرار while مرة أخرى في هذا الملف

```
while($row = mysql_fetch_object($result1))
```

هنا قمنا بخلق المتغير row ولا تنس في php ليس من الضروري تعريف المتغيرات قبل استخدامها وقمنا بمساواته (وهو الشرط الخاص بأمر التكرار while) بالدالة mysql_fetch_object مسندين إليه قيمة المتغير result1

الدالة mysql_fetch_object وظيفته أو من ضمن الوظائف التي تقوم بها أنها تقوم بانتزاع وعرض نتائج الاستعلام.

فإذا تحقق شرط أمر التكرار while فستنفذ هذا الكود الطويل قليلاً.

```
while($row = mysql_fetch_object($result1)) {
echo "<td align=middle>SiteNmae<br><input type=text name='name'
value='$row->sitename'></td>";
echo "<td align=middle>Add Date<br><input type=text name='add_da
te' value='$row->add_date'></td>";
echo "<td align=middle>Email<br><input type=text name='email' value
='$row->email'></td>";
echo "<td align=middle>SiteUrl<br><input type=text name='siteurl' val
ue='$row->site_url'></td></tr>";
echo "<td align=middle>Description<br><input type=text name='desc'
value='$row->description'></td></tr>";
$id = $row->id;
}
echo "</table><p>";
echo "<br><input type=hidden name='id' value='$id'>";
?>

<input type=submit name="update" value="Edit The Info">
</form>
</tr>
</table>
<?
if ($update) {
if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $
desc == "") {
die("<b>You left one or more fields blank.</b>");
}

add_date='$add_date'،    $query2 = "update dalal set sitename='$name'
description='$desc' ، site_url='$siteurl'، email='$email'،
where id='$id'";
$query2);،     mysql_db_query('links'
echo "<b>Your record has been updated</b></p>";
}
mysql_close();
?>
```

هنا قمنا بطباعة المعلومات والتحقق بواسطة شرط IF دعنا نأخذ عملية طباعة المعلومات:

```
echo "<td align=middle>SiteNmae<br><input type=text name='name' value='$row->sitename'></td>";
echo "<td align=middle>Add Date<br><input type=text name='add_date' value='$row->add_date'></td>";
echo "<td align=middle>Email<br><input type=text name='email' value='$row->email'></td>";
echo "<td align=middle>SiteUrl<br><input type=text name='siteurl' value='$row->site_url'></td></tr>";
echo "<td align=middle>Description<br><input type=text name='desc' value='$row->description'></td></tr>";
$id = $row->id;
```

هنا في أول سطر قام بطباعة كود HTML وبداخله أيضاً بعض الأشياء من PHP لا تخف فالمقصود هنا أنه قام بتخزين جميع ما يطبعه بداخل المتغير الخاص بلغةHTML فهنا مثلاً

```
echo "<td align=middle>SiteNmae<br><input type=text
name='name' value='$row->sitename'></td>";
```

قام بطباعة كود HTML وهو عبارة عن حقل الإدخال (INPUT) من نوع (TEXT) وقام بترميزه إلى الحقل الموجود بقاعدة البيانات من المتغير row وتخزينه في قيمة المتغير في html (name)

والأكثر وضوحاً ما قمنا به في السطر السابق قمنا بوضع المتغير row ثم يليه اسم الحقل في الجدول وهنا اسم الحقل sitename.

أظن مع تشغيلك للسكريبت سوف تتضح أكثر وأكثر وهذا الذي استخدمته مع الجميع

بعد ذلك قمت بطباعة أكواد html وهي عبارة عن فورم للإدخال ويأتي بعده شرط if وهو هنا

```
<input type=submit name="update" value="Edit The Info">
</form>
</tr>
</table>
<?
if ($update) {
if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $desc == "") {
```

```
die("<b>You left one or more fields blank.</b>");
}

add_date='$add_date'،   $query2 = "update dalal set sitename='$name'
description='$desc' ، site_url='$siteurl'، email='$email'،
where id='$id'";
$query2);،    mysql_db_query('links'
echo "<b>Your record has been updated</b></p>";
}
mysql_close();
?>
```

في الشرط نريد التأكد هل تم تعديل أحد البيانات أم لا وهو شبيه بالشرط الذي استخدمته مع ملف add.php بعده إذا كان هناك إدخال فسيقوم بعدة أوامر وأولها شرط آخر if يتأكد هل هناك من الحقول التي جعلت فارغة أم لا وهي هنا:

```
if($name == "" || $add_date == "" || $email == "" || $siteurl == "" || $
desc == "") {
die("<b>You left one or more fields blank.</b>");
}
```

إذا كان هناك حقول فارغة سوف يطبع له You left one or more fields blank أي أنك تركت حقل أو أكثر فارغ.

بعد ذلك يأتي هذا:

```
،$query2 = "update dalal set sitename='$name'
، site_url='$siteurl'، email='$email'، add_date='$add_date'
description='$desc'
where id='$id'";
$query2);،    mysql_db_query('links'
echo "<b>Your record has been updated</b></p>";
}
mysql_close();
```

قمنا هنا في أول سطر بتعريف المتغير query2 بعد ذلك خزنا بداخله قيمة أمر SQL التحديث update .

أمر update يقوم بتحديث المعلومات في قواعد البيانات من تجديد وحذف.

ويكتب هكذا :

Update tablename Set (values

الكلمات التي تبدأ بحروف كبيرة هي الثوابت بمعنى لا يتم تغييرها والتي تبدأ بحروف صغيرة مثل sitename و values فهي التي يتم تغييرها كما وضحت في الكود السابق فقد قمت بمساوات المتغيرات description،site_url،email،add_date،name بما يقابلها من الحقول في الجدول dalal ووضعت شرط where أن يكون المتغير
id=id$

وفي السطر الذي يليه هذا قمت باستخدام الدالة mysql_db_query مسنداً إليه اسم قاعدة البيانات والمتغير الذي خزن قيمة التحديث وهو query2 لكي يتم تحديث قاعدة البيانات
وفي السطر الذي يليهما استعملت الدالة mysql_colse لكي يتم إغلاق الاتصال بقاعدة البيانات
وهو آخر سطر اليوم

السكريبت يتكون من ثلاثة ملفات، الأول view.php لعرض المدخلات في سجل الزوار، والثاني add.php لإضافة مدخل جديد إلى سجل الزوار، والثالث هو config.php ويحتوي على بيانات قاعدة البيانات.
في الملف view.php، يتم جلب البيانات المدخلة من قاعدة البيانات وعرضها واحدة تلو الأخرى في صورة جدول HTML، وأسفل هذا الجدول يوجد نموذج لإضافة تعليق جديد في سجل الزوار.
الملف add.php يقوم بأخذ البيانات المرسلة من النموذج الموجود في الملف view.php، ثم يقوم بمراجعة هذه البيانات والتحقق منها، ثم إضافتها إلى قاعدة البيانات وإعادة المستخدم إلى الملف view.php.

يجب أن نحدد الآن البيانات التي نريد تخزينها في قاعدة البيانات:

- الاسم.
- البريد الإلكتروني.
- الصفحة الشخصية.
- التعليق على الموقع.

❖ **قاعدة البيانات**

من خلال المعلومات السابقة نلاحظ بأننا سنحتاج إلى جدول واحد فقط في قاعدة البيانات، وليكن اسمه guestbook، هذه هي الشيفرة التي يجب تنفيذها للحصول على البنية الأساسية للجدول:

```
CREATE TABLE guestbook (
،    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
،    name CHAR(100)
،    email CHAR(128)
،    homepage CHAR(200)
،    date DATETIME
،    ip CHAR(15)
message TEXT
);
```

شيفرة SQL السابقة تعرف جدولاً اسمه guestbook يحتوي على الحقول التالية:

id - هذا الحقل يخزن رقماً تسلسلياً يزداد بمقدار واحد مع كل حقل جديد يضاف إلى الجدول، ومواصفاته:

رقم INTEGER

بلا إشارة (موجب دائماً) UNSIGNED

إجباري (لا يمكن أن يترك خالياً) NOT NULL

يتم تحديثه تلقائياً إلى رقم أكبر من السابق بواحد AUTO_INCREMENT

وهو المفتاح الأساسي للوصول إلى بيانات الجدول PRIMARY KEY

name - هـذا الحقـل يخـزن اسـم الـشخص الموقـع في سـجل الـزوار، ومواصـفاته كالتالي:
يخزن نصاً بطول ١٠٠ حرف كحد أقصى CHAR (100)

email - لتخزين البريد الإلكتروني، ١٢٨ حرفاً كحد أقصى (هـذا الـرقم قياسـي تقريباً في أغلب البرامج القائمة على الويب لتخزين عناوين البريد الإلكترونية).

homepage - لتخزين عنوان الصفحة الشخصية للموقع، ٢٠٠ حرفاً بحد أقصى -- قيمة معقولة.
date - لتخزين تاريخ المدخل في سجل الزوار، وهو يخزن قيماً من النوع: تاريخ ووقت DATETIME

message - لتخزين نص الرسـالة (تعليقـك علـى الموقـع)، وهـو حقـل يتـسع لتخـزين عدد كبير جداً من الحروف TEXT

❖ الملف config.php

يحتـوي علـى الملـف علـى شـيفرة برمجيـة تقـوم بإعـداد المـتغيرات اللازمـة للوصـول إلى قاعدة البيانات، وسنقوم بإضافة شيفرة الاتصال بقاعدة البيانات إلى هذا الملف بـدلاً من تكرارها في كل من الملفين view.php و add.php.
أولاً سنقوم بإعداد مجموعة من المتغيرات:

```
/* store the hostname of the MySQL server */
$dbhost = "localhost";
/* store the username to login to MySQL */
$dbuser = "root";
/* store the password to login to MySQL */
$dbpass = "";
/* store the name of the MySQL database */
$dbname = "";
```

كما تلاحظ، قمنا بتعريف أربعة متغيرات مختلفة هي:

dbhost$ -- وتقوم بتخزين عنوان مزيد قاعدة البيانات.

dbuser$ -- وتقوم بتخزين اسم المستخدم الذي سيستخدم للدخول إلى قاعدة البيانات.

dbpass$ -- وتقوم بتخزين كلمة المرور التي ستستخدم مع اسم المستخدم للدخول إلى قاعدة البيانات.

dbname$ -- تخزن اسم قاعدة البيانات التي ستحوي الجدول guestbook.

يمكن للمستخدم الآن تعديل معلومات هذه المتغيرة عن طريق فتح الملف في المفكرة مثلاً وتعديل القيم الموجودة بين علامات الاقتباس مقابل أسماء المتغيرات.

بعد تعريف المتغيرات السابقة، سنقوم كما اتفقنا بوضع الشيفرة اللازمة للاتصال بقاعدة البيانات، وهي عبارة عن استدعاء لدالتين:

```
/* connect to the MySQL server */
$dbpass);،$dbuser،mysql_connect($dbhost
/* set $dbname as the database to be used */
mysql_select_db($dbname);
```

❖ **شرح الدوال**

```
password)، username،mysql_connect(hostname
```

تقوم الدالة السابقة بالاتصال بمزود قاعدة البيانات الموجود على العنوان hostname، وتحاول الدخول إلى المزود قاعدة البيانات عن طريق إرسال اسم المستخدم username و كلمة المرور password، في حال نجاح العملية تعيد مورداً (resource) نستطيع استخدامه مع دوال أخرى لتنفيذ أوامر قاعدة البيانات على هذا الاتصال المفتوح (يمكن فتح أكثر من اتصال بأكثر من قاعدة بيانات في نفس الوقت).

جميع المتغيرات الممررة إلى الدالة اختبارية، إذا لم تدخل password فإن الدالة ستحاول إجراء الاتصال دون تمرير كلمة مرور، وإذا لم تدخل username فستحاول الدالة إجراء الاتصال دون اسم مستخدم، وإذا لم تحدد hostname فإن الدالة ستحاول الاتصال بقاعدة البيانات على العنوان المحلي (locahost).

إذا كنت قد ركبت MySQL مباشرة دون تغيير الإعدادات فإن مزود قواعد البيانات سيسمح مباشرة بالاتصال من الجهاز المحلي باسم المستخدم root والذي لن تكون له كلمة مرور افتراضية، وبالتالي يمكنك الاتصال مباشرة إلى مزود MySQL المحلي عن طريق القيم الموضحة في المتغيرات التي في الأعلى.

```
link_identifier)،mysql_select_db(database_name
```

تقوم هذه الدالة بتحديد قاعدة البيانات التي سيتم العمل عليها، حيث أن مزود قاعدة البيانات يمكن أن يحتوي على أكثر من قاعدة بيانات واحدة.

تحدد اسم قاعدة البيانات عن طريق المتغيرة database_name، أما المتغيرة الثانية فهي الاتصال الذي تريد أن تحدد قاعدة البيانات له، فإذا كنت قد خزنت القيمة التي أعادتها الدالة mysql_connect في متغيرة، وأردت الآن تعيين قاعدة البيانات الذي يجب استخدامها في ذلك الاتصال، فيمكنك تمرير المتغيرة إلى الدالة mysql_select_db في الوسيطة الثانية، ولكن هذه الوسيطة الثانية اختيارية، فإذا لم تقم بتمريرها فإن الدالة ستعيد قاعدة البيانات التي يجب العمل عليها لآخر اتصال تم فتحه بقاعدة البيانات، وإذا لم يكن هناك اتصال مفتوح من قبل ستحاول الدالة فتح اتصال بقاعدة البيانات كما لو كنت قد شغلت ()mysql_connect بدون تحديد الـ hostname و الـ username والـ password وثم تعيين قاعدة البيانات التي يجب العمل عليها لهذا الاتصال الجديد.

في ملفنا المثال قمنا بتنفيذ هذه الدالة مباشرة بعد فتح اتصال بقاعدة البيانات باستخدام الدالة mysql_connect، وبالتالي عينا قاعدة البيانات dbname$ كقاعدة البيانات التي يجب استخدامها مع آخر اتصال فتح بقاعدة البيانات.

هذه الدالة تغير true (صح) في حالة نجاحها و false (خطأ) في حالة فشلها.

يمكننا الآن النظر إلى نسخة نهائية كاملة من الملف config.php:

```
<?php
/* store the hostname of the MySQL server */
$dbhost = "localhost";
/* store the username to login to MySQL */
$dbuser = "root";
/* store the password to login to MySQL */
$dbpass = "";
/* store the name of the MySQL database */
$dbname = "";
/* connect to the MySQL server */
$dbpass);،$dbuser،mysql_connect($dbhost
/* set $dbname as the database to be used */
mysql_select_db($dbname);
?>
```

❖ الملف view.php:

في البداية المقدمات العادية ورأس المستند بلغة HTML:

```
<html dir="rtl">
<head>
</title>سجل الزوار<title>
<style>
textarea {،select،input،td،body
font-family: Tahoma;
font-size: x-small;
}
h6 {،h5،h4،h3،h2،h1
font-family: Arial;
}
</style>
</head>
<body>
</h1>سجل الزوار<h1>
```

بعد أن تتم طباعة رأس المستند بلغة HTML نأتي إلى الجزء التالي وهو جلب البيانات من قاعدة البيانات وعرضها للمستخدم.

في البداية نحتاج إلى اتصال بقاعدة البيانات، وكما قلنا من قبل، الملف config.php يحتوي على الشيفرة الكاملة لتجهيز اتصال بقاعدة البيانات، ما علينا القيام به الآن هو إعطاء أمر بتنفيذ البيانات الموجودة في الملف config.php، وهذا الأمر هو include، كالتالي:

```
/* get a database connection */
include 'config.php';
```

النتيجة التي يعطيها الأمر include السابق هي نفسها النتيجة التي كنا سنحصل عليها لو كنا وضعنا الشيفرة الموجودة في الملف view.php مباشرة، بمعنى أننا بدلاً من أن ننسخ شيفرة قاعدة البيانات الموجودة في الملف config.php يدوياً ونضعها في أعلى الملف view.php للحصول على وصول إلى قواعد البيانات، فإننا نطلب من مترجم PHP أن يقوم بالعملية آلياً، حيث نطلب منه تنفيذ الشيفرة الموجودة في الملف config.php كما لو كانت جزءاً من الملف الذي استدعى الأمر (view.php).

❖ الأمر **include**:

يستخدم الأمر include كالتالي:

```
include filename;
```

أي الكلمة include وبعدها اسم الملف الذي نريد إدراجه، وفي مثالنا كان الملف هو config.php.

❖ **طلب البيانات من قاعدة البيانات**

المفترض الآن بأن جميع الإضافات التي تمت إلى سجل الزوار مخزنة في الجدول guestbook في قاعدة البيانات، وأن لدينا اتصالاً الآن بقاعدة البيانات من خلال الملف config.php، نستطيع البدأ بعمل استعلام لقاعدة البيانات نطلب فيه البيانات التي نريدها، ويكون ذلك بالشيفرة التالية:

```
$res = mysql_query("SELECT
،homepage،email،name
،UNIX_TIMESTAMP(date) AS date
```

```
message
FROM guestbook
ORDER BY date DESC");
```

الدالة **mysql_query**:

```
mysql_query(query، link_identifier);
```

تقوم الدالة بإرسال الاستعلام (query) إلى قاعدة البيانات عبر الاتصال الذي يمرر إليها (link_identifier)، وإذا لم يكن هنالك اتصال ممرر، فإن الدالة تستخدم آخر اتصال فتح بقاعدة البيانات، وإذا لم يكن هنالك اتصال قد فتح من قبل ستحاول الدالة إجراء اتصال بقاعدة البيانات عن طريق تنفيذ الدالة mysql_connect وتمرر إليها قيم hostname و username و password فارغة.

في مثالنا السابق، قمنا بتمرير قيمة واحدة فقط إلى الدالة هي الاستعلام الطويل المحاط بعلامتي الاقتباس " و "، وستقوم الدالة باستخدام آخر اتصال فتح بقاعدة البيانات وهو الاتصال الذي فتح في الملف config.php الذي أدرجناه من قبل في ملفنا (view.php).

نأتي الآن إلى الاستعلام، وهو عبارة عن استعلام بسيط من النوع SELECT كتب بلغة SQL. وسأقدم شرحاً سريعاً هنا.

SELECT: يعني اختيار أو جلب.

```
name، email، homepage، UNIX_TIMESTAMP(date) AS date، message
```

هذه هي أسماء الحقول التي طلبناها (جلبناها) من الجدول، وهي عبارة عن قائمة بسيطة من أسماء الحقول، ما عدا التالي:

```
UNIX_TIMESTAMP(date) AS date
```

تقوم الدالة UNIX_TIMESTAMP بإعادة التاريخ المخزن في قاعدة البيانات في صورة TIMESTAMP الخاص بنظام unix ، وهو عبارة عن نظام للتاريخ يحسب عدد الثواني منذ منتصف الليل في ١٩٧٠/١/١ ، وبالتالي فإن هذه القيمة تتغير في كل ثانية ، تعتبر الـ Timestamp الوحدة القياسية لتخزين المعلومات عن التاريخ والوقت في PHP وأغلب لغات برمجة ونظم Unix ، وبالتالي فإننا طلبنا من قاعدة البيانات أن تحول لنا قيمة التاريخ (date) إلى الـ Timestamp المقابل حتى يسهل التعامل معها في PHP وعرضها بالطريقة التي نريدها..

بعد ذلك أضفنا المقطع AS date ، هذا المقطع يعني بأن الحقل هذا يجب أن يسمى بالاسم date ، السبب في إضافتنا لهذا المقطع هي أن الحقول تسمى عادة بأسماء التعبيرات التي تتكون منها ، فهذا الحقل مثلاً :

```
UNIX_TIMESTAMP(date)
```

سيكون اسمه هو :

```
UNIX_TIMESTAMP(date)
```

وهذا سيجعل التعامل معه في PHP صعباً نوعاً ما ، لذا قلنا لقاعدة البيانات أن تسمي البيانات هذه بالاسم date بدلاً من الاسم السابق.

بعد الانتهاء من قائمة الحقول نجد المقطع التالي:

```
FROM guestbook
```

FROM

الكلمة FROM تعني (من) ونحدد بعدها اسم الجدول الذي نريد طلب (جلب) البيانات منه ، وهو guestbook في مثالنا السابق.

```
ORDER BY date DESC
```

المقطع السابق يحدد الطريقة التي ترتب بها البيانات..

ORDER BY

تعني (ترتيب حسب) وهي تحدد الحقل الذي نريد الترتيب تبعاً له، وفي مثال الحقل date، أي أننا نريد ترتيب المدخلات حسب تاريخ إضافتها، ولكن استخدام ORDER BY وبعده اسم الحقل يعني فرز البيانات حسب ذلك الحقل تصاعدياً، أي من الأقل إلى الأكبر، ولكننا في الواقع نريد فرزها في برنامجنا تنازلياً (من الأحدث أو الأعلى تاريخاً إلى الأقدم) ولهذا السبب أضفنا المقطع DESC وهو اختصار لكلمة descending أي تنازلياً.

انتهى

الفهرس